A SURVEY OF ZERO-KNOWLEDGE TECHNIQUES AND THEIR APPLICATIONS

BY

SUNIL KUMAR ROTTOO

A Thesis

Submitted to the Faculty of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Computer Science

University of Manitoba

Winnipeg, Manitoba

ISBN 0-315-77027-9

A SURVEY OF ZERO-KNOWLEDGE TECHNIQUES AND THEIR APPLICATIONS

BY

SUNIL KUMAR ROTTOO

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Abstract

Zero-knowledge protocols were introduced in 1985 and have since been the subject of extensive research. Such protocols have had significant theoretical and practical ramifications for the field of cryptography. The basic idea is as follows: a prover (Peggy) owns a secret and through a series of exchanged messages, wishes to convince a verifier (Vic) that she knows the secret without revealing anything about it (beyond the fact that she knows it.) Also, if Peggy does not know the secret and attempts to cheat, Vic will almost certainly detect the cheating. Furthermore, Vic cannot convince anyone else of the existence of Peggy's secret by exhibiting a transcript of his conversation with Peggy.

Since the field of zero-knowledge is a relatively young one, this thesis is an attempt to provide a broad survey of the area as well as to present a somewhat in-depth examination of the more important ideas that have emerged so far. The basic concepts are first discussed in some detail, and some of the more important zero-knowledge protocols are then presented, with examples taken mainly from number theory and graph theory. These are then generalized and different methods for building zero-knowledge protocols for all languages in NP are presented. The latter are compared and evaluated in terms of efficiency and suitability for practical implementation.

At the heart of zero-knowledge protocols lies the principle of bit commitment. The different levels of security obtained by specific implementations of bit commitment protocols are given. A very important result, namely the concept of non-interactive zero-knowledge protocols is also discussed. Finally, some of the cryptographic applications of both interactive and non-interactive zero-knowledge protocols are presented and directions for future research in the area are suggested.

Acknowledgements

I would like to thank the following individuals for helping both directly and indirectly with the thesis: my supervisor, Hugh Williams, for suggesting a fascinating thesis topic and for putting up with my interest swings that would give whiplash to any normal human being; Neil Arnason for his generosity in paying me probably more than I deserved when I worked for him; Heidi Arnason for many stimulating discussions about the meaning of life; Renate Scheidler for her friendship and proofreading my thesis (only a real friend would willingly submit to such an exercise); and finally my mother, for teaching me that perseverance pays off.

Table of Contents

Chapter 1
Introduction to Zero-knowledge

§1.1 Introduction

The concepts of *interactive proofs* and *zero-knowledge* were introduced by Goldwasser, Micali and Rackhoff [GMR1] in 1985 and have since been the subject of extensive research ([GRM1],[BCC],[GQ1],[FFS],etc). Although interesting for their theoretical implications, these concepts have been shown to be applicable to cryptographic protocols. In Chapter 6 we shall given a brief of some of the contributions of zero-knowledge to the field of cryptography.

Informally an *interactive proof system* is a two-party conversation during which an infinitely powerful prover (P) attempts to convince a polynomial-time verifier (V) of some fact, usually of the form $x \in L$ where L is some language, through a series of exchanged messages. It should be emphasized that these proofs are not proofs in the mathematical sense of the word but in a probabilistic sense as there is always a (small) theoretical possibility of the verifier being convinced of the validity of a false theorem. Such an interactive proof is said to be zero-knowledge if, whenever the verifier is told by a trusted oracle that $x \in L$, he/she is able to compute on his/her own the conversations that he/she would have had with the prover during an interactive proof of $x \in L$. If the end result of a protocol between two interactive parties P and V is to have P transmit to V the value of some predicate (in this case the value of $x \in L$) then we should require that the protocol reveal nothing more than this value. Furthermore, the above should also hold even if V tries to cheat. So the verifier does not learn anything new from the prover other than what the prover had set out to prove. Throughout this thesis we will informally refer to the prover in a protocol as Peggy and to the verifier as Vic. Symbolically the following notation will be used-

P(V) refers to the real prover (verifier) who is honest and follows the protocol.

$P^*(V^*)$ refers to an arbitrary (possibly cheating) prover (verifier) who is allowed to deviate arbitrarily from the protocol.

By varying the limits we place on the computational abilities of our interacting parties we can have different models of protocols. Specifically we allow the prover or the verifier to be machines with infinite power or we restrict them to polynomial-time computations.

To give a concrete example, consider Peggy who is trying to convince Vic that a certain integer x is a quadratic residue mod N where N is the product of two primes known only to Peggy. If the protocol which is used by both parties is to be zero-knowledge, Vic should not be able to trick Peggy into revealing a square root of x or the factors of N nor any information that would allow Vic to compute these things much faster than before the start of the protocol. More generally, the data that Vic sees during an execution of the protocol give rise to a probability distribution which depends on the initial input. We can thus define zero-knowledge to mean that given the fact that $x \in L$, Vic can generate on his own and without any help from Peggy, a simulation of the conversation that he would have had with Peggy that is indistinguishable from the one that might have been generated during a real conversation with Peggy.

In order to illustrate in a very intuitive manner the basic feature of zero-knowledge protocols we present and adaptation of the following narrative due to Guillou and Quisquater and [GQ3].

§1.2 Ali Baba and the 40 thieves Revisited

The story begins in 9th century Baghdad where Ali Baba was visiting the Old Market Place. Suddenly, a thief grabbed his purse and ran off. Ali Baba followed the thief and saw him enter a cave whose plan is reproduced in Figure 1. Unfortunately, Ali Baba did not see which of the passages the thief had taken. So he took the right passage, looking for the thief along the way. Ali Baba reached a dead end without finding the thief. He decided

to go back and explore the other passage and again reached a dead end without finding the thief. Ali Baba was puzzled as to where the thief had gone but he left it at that.

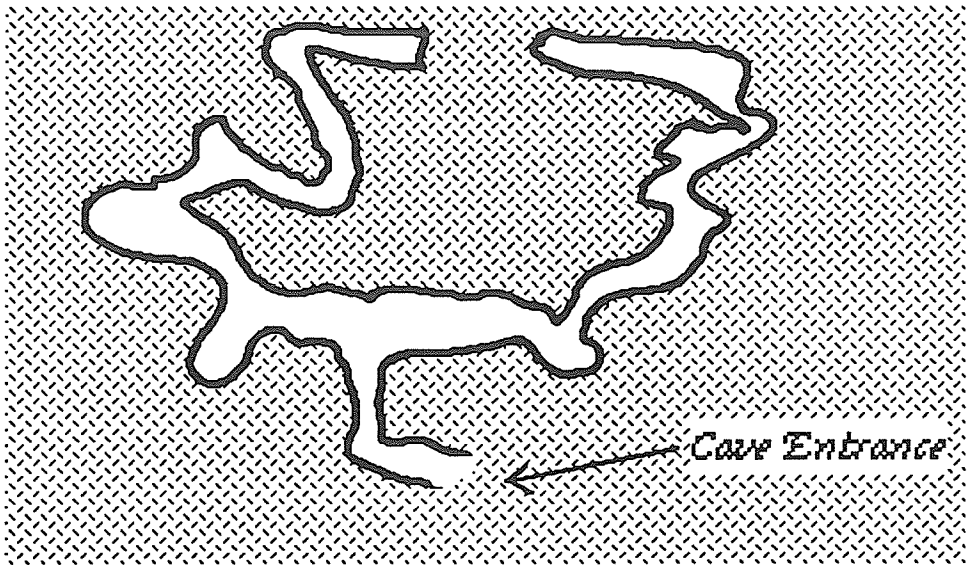

Figure 1

The next day at the market another thief stole Ali Baba's basket and found refuge in the same cave. As before, Ali Baba did not see which way the thief went. He decided to explore the left passage and could not find the thief. After looking along the other passage he again came up empty-handed. Ali Baba reasoned that in both cases the thieves were lucky in choosing the passage that Ali Baba chose not to explore first and escaped while he was exploring the wrong passage. Days went by and every day Ali Baba was robbed by a different thief. And thus, the strange ritual of running after the thief, picking a random passage and not finding the thief was repeated until the 40th day when Ali Baba finally realised that there was something more to the cave. Indeed, the probability that he would always pick the wrong passage 40 times in succession was $1/2^{40}$, which is VERY small. Ali Baba resolved to discover the secret of the cave. The next day, instead of going to the market, Ali Baba hid beneath sacks at the end of one of the passages. After a long

uncomfortable wait he saw a thief arriving, who sensing that he was being pursued by his victim whispered the magic words "Open Sesame!", whereupon the wall of the cave slid open and the thief ran through the opening. The wall then closed leaving a disappointed pursuer staring at a blank wall. After the pursuer had left, Ali Baba experimented with the magic words and after walking through the newly opened passage discovered that the two passages were in fact connected as shown in Figure 2. Ali Baba had discovered the secret of the cave and at the same time understood how the forty thieves had escaped from him. After experimenting with the magic words Ali Baba managed to change them to something else so that the wall would now no longer work with the words "Open Sesame!". The very next day a thief was caught. Ali Baba wrote down an encrypted version of the magic words on a manuscript. After his death the manuscript was lost until it was rediscovered by researchers at M.I.T. in 1985.

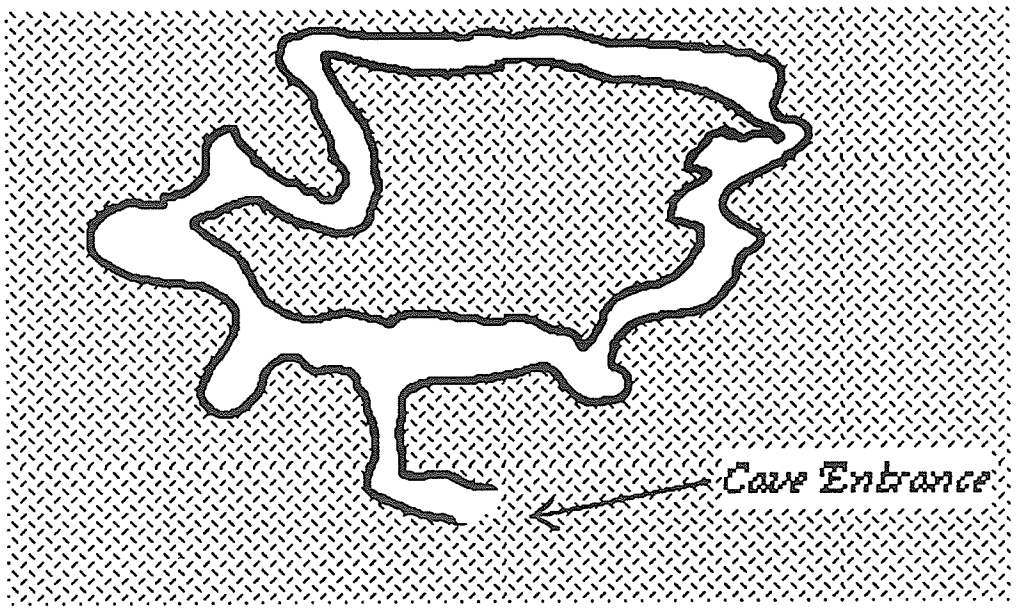

Figure 2

The researchers managed to decipher the code and found the magic words. Archaeological excavations in Baghdad revealed the cave and it was found that the magic words still

worked. A member of the team of researchers, a certain Mick Ali who was also a descendent of Ali Baba wanted to demonstrate that he knew the secret but he did not wish to reveal it. This is what he did.

CNN (always in Baghdad) was granted an exclusive to the story and a television crew filmed a detailed tour of the cave with the two dead-ends. Then everybody went out and Mick went in alone and went down one of the passages. A reporter (Peter A.) then went in with a rolling camera but only as far as the fork. He then flipped a coin. If the coin came up heads (tails) he would ask Mick to come out on the right (left). The coin came up heads and Mick was asked to come out on the right which he did. This scene was repeated 40 times, each time the crew filming the whole cave, everyone then leaving, Mick going in alone and down one of the passages, and finally Peter asking to come out the appropriate passage as determined by the outcome of his coin toss. Of course, since Mick knew the secret he succeeded in all 40 scenes. Anyone who did not know the secret would have been caught on the first failure. A cheater is detected when the reporter asks him to come out on the right when he went down the left passage or vice-versa. Each new test thus reduces by a factor of half the probability that the reporter will be fooled. By increasing the number of tests the reporter can increase his confidence that the prover is not cheating.

The Simulated Tape

Another T.V. reporter (Dan R.) hired by a rival network also wanted to film the story but Mick had given exclusive rights to CNN and could not participate. However, he suggested to Dan that the story could still be filmed *without knowing the secret.* After A LOT of thought Dan finally understood how it could be done. He hired a stage actor who looked like Mick and went through the same procedure that Peter and Mick had gone through. Of course, during the course of filming about half of the scenes were spoilt as Mick's double did not know how to get from one passage to the next. However, Dan kept filming scenes until he had forty successful ones. The spoilt scenes were edited out. That night, both

tapes were broadcast on each network. The matter was quickly taken to court with the two tapes being the evidence. However, after viewing both tapes the judges could not tell the genuine tape from the simulated one. Clearly the simulated tape did not reveal anything about the secret as the double did not know the secret. But the simulated and genuine tape were indistinguishable. Therefore the genuine tape also did not convey anything about the secret. Recall that at the time Peter A. had been convinced that Mick knew the secret but he could not transfer his confidence to the judges or the television audience.

And by this demonstration Mick had shown that one could "convince without revealing" which is the hallmark of zero-knowledge protocols.

§1.3 Some Complexity Theory

In the next chapter we will rigorously define the concept of interactive and zero-knowledge proofs. In order to understand these on a more intuitive level, it is recommended that the reader attempt to relate them to the rather informal narrative given above. In this section we will give a brief overview of the following well-known concepts from the theory of computation.

A function $f(n)$ is expressed in the form $O(g(n))$ (called the "big-O" notation) where $f(n) = O(g(n))$ if there exists constants c and n_0 such that

$$f(n) \leq c \mid g(n) \mid \text{ for } n \geq n_0.$$

For example, if $f(n) = 6n + 8$, then $f(n) = O(n)$ since $6n + 8 \leq 7n$ for $n \geq n_0 = 8$ and $c = 7$.

The computational complexity of an algorithm A, which takes as input some x ($| x | = n$)[1] is measured by its time and space requirements, $T(n)$ and $S(n)$. The O-notation allows the classification of any algorithm in terms of $T(n)$ and $S(n)$. For example, an analysis of A might reveal that $T(n) = an^2 + bn + c$ where $a,b,$ and c are unspecified constants. Since n^2

---

[1]Note that in the above equation, the | |-notation was used to refer to absolute value. From now onwards, and unless otherwise specified, the notation |x| refers to the number of bits required to represent x.

is the dominant term we say that $T(n) = O(n^2)$. Thus an algorithm is said to be *polynomial-time* if its running time is given by $T(n) = O(n^t)$ for some constant t. It is *exponential-time* if $T(n) = O(t^{p(n)})$ for some constant t and polynomial $p(n)$. It should be noted that any problem is classified according to the *minimum* space and time required to solve the hardest instance(s) of the problem on a Turing machine. Informally, a Turing machine can be thought of as an idealized computer. Its purpose is to standardize ideas of computability and computation time by referring all problems to one standard machine. A Turing machine can also be considered as a "realistic" model of computation in that problems that are solvable on a Turing machine are also solvable on real systems and vice-versa. Such problems are considered to be tractable as they can usually be solved for reasonably sized inputs.

The class **P** consists of all problems solvable in polynomial-time. The class **NP** (nondeterministic polynomial) consists of all problems that can be solved in polynomial time on a non deterministic Turing machine. By this, we mean that the machine will guess the solution and can then check its correctness in polynomial-time. Of course, there is no guarantee that the machine will guess the right answer. It is clear that the class **NP** includes the class **P** as any problem that is polynomially solvable on a deterministic Turing machine is also polynomially solvable on a nondeterministic one. It would seem that to *deterministically* solve certain problems in **NP** requires exponential time and these problems are thus considered intractable. A famous example of such a problem is the circuit satisfiability problem where one is required to determine whether there exist assigned values to a set of n Boolean variables $P = P_1,P_2,...,P_n$ such that a given collection of clauses over P is true. Cook has shown that every problem in **NP** can be reduced, in polynomial time, to a satisfiability problem. Thus, if the satisfiability problem is polynomially solvable, then every problem in **NP** is polynomially solvable and if some problem in **NP** is intractable, then the satisfiability problem is intractable. The set of all problems that can be shown to be equivalent to satisfiability is called the set of **NP**-

complete problems and if any one of the NP-complete problems were found to be in P, all NP problems would be in P. As such, NP-complete problems are the hardest problems in NP with the property that known algorithms for solving them have worst-case complexities that are exponential in the size of the input. The class co-NP consists of all problems that are the complements of some problems in NP. Intuitively, an NP problem is a question of the form "determine whether a solution exists " whereas a co-NP problem can be formulated as "show that there are no solutions." As a matter of convenience the theory of NP-completeness is designed to be applied only to *decision problems*. These are problems which have only two possible solutions, "yes" or "no." For example, in the satisfiability problem described above, the problem could be formulated to read: "Does there exist a set of n Boolean variables such that a given collections of clauses over P is true?"

The concepts that are outlined above will be used extensively throughout this thesis and should be sufficient for our purposes. For a more extensive discussion of these concepts the reader is referred to [GJ].

§1.4 Random Variables and Probability Distributions

Since these two concepts are central to the definition of zero-knowledge, we will now give a brief introduction and definitions. Informally, a random variable is one whose observed value is determined by chance. Random variables usually fall in two categories; they are either discrete or continuous. For example a random variable T can represent the time of day at which demand for electricity peaks. This is a random variable since its value is affected by chance factors such as time of the year, humidity, and temperature. Thus T can take on any value in the 24-hour time span. If the number of possible values a random variable can take on is finite or a countably infinite set, then it is a discrete random variable; otherwise it a continuous random variable. Thus, since time is measured continuously the

variable T above is a continuous random variable. Throughout this thesis we will be using the idea of discrete random variables and we now give a formal definition.

Definition. A discrete random variables is one which can assume at most a finite or a countably infinite set of values.

However, when dealing with random variables it is not enough just to determine what values are possible. We also need to determine what is probable. That is, we want to be able to predict what the values that the variable is likely to assume at any given time. Thus, we want to be able to assign probabilities to each of the possible values that the random variable can take on. Such a set of probabilities describes what is known as the *probability distribution.*

Chapter 2
Zero-knowledge Defined

§2.1 Introduction

As mentioned in the previous chapter, we will now give the required definitions that will be used throughout this thesis.

§2.2 Interactive Turing Machines and Protocols and Proof Systems

Definition [GMR2] : An *interactive Turing Machine* (ITM) is a Turing machine with a read-only input tape, a read-only random tape, a read/write work-tape, a read-only communication tape, a write-only communication tape and a write-only output tape. The random tape contains an infinite sequence of random bits which can be thought of as unbiased coin tosses and can be scanned only from left to right. Thus the machine flips a coin by reading the next bit from its random tape. The contents of the write-only tape can be thought of as messages sent by the machine while those of the read-only tape can be thought of as messages received.

Definition [GMR2] : An *interactive protocol* is an ordered pair of ITM's (P,V) which share the same input tape. V's write-only communication tape is P's read-only communication tape and vice-versa. Machine P is not computationally bounded while Machine V's computation is bounded by a polynomial in the length of the common input. The two machines take turns being active, with V starting first. During an active stage, Machine P (V) first performs some internal computation based on the contents of its tapes and then writes a string for V (P) on its write-only communication tape. As soon as machine P (V) writes its message it is deactivated and unless the protocol is over, Machine V (P) now becomes active. Either machine can terminate the protocol by not sending any messages during its active phase. Machine V accepts (is convinced of the validity of the proof) or rejects by entering an accept or reject state thereby stopping the protocol.

Definition [GMR2] : let L be a language over $\{0,1\}^*$. Let (P,V) be an interactive protocol where P is the prover and V the verifier. We say that (P,V) is an **Interactive Proof System** for L if the following conditions hold:

1. Completeness (in a probabilistic sense). For any $x \in L$ given as input to (P,V), V halts and accepts with probability at least $1 - |x|^{-k}$ for sufficiently large x, where k is some constant greater than 0.
2. Soundness (in a probabilistic sense). For any ITM P^* which can interact with V, and for sufficiently large $x \notin L$, V accepts with probability at most $|x|^{-k}$.

   where k is some constant greater than 0.

Informally, condition (1) says that if $x \in L$, then V will accept with high probability while condition (2) ensures that if $x \notin L$ then it is highly unlikely that even a cheating machine will be able to convince V of the veracity of the statement $x \in L$. In fact V only has to trust the randomness and secrecy of his own coin tosses.

Definition: We define **IP**, Interactive Polynomial-time, to be the class of languages for which there exist interactive proof systems.

§2.3 Zero-Knowledge

Intuitively, a zero-knowledge proof is one which reveals only its validity. Recall from §1.1 that we referred to zero-knowledge as the ability of the verifier to generate on his/her own, after being told by a trusted third party that the prover's assertion is correct, a probability distribution that is indistinguishable from (or reasonably close to) that generated during a conversation with the prover. We next focus on the notion of indistinguishability for random variables.

§2.3.1 Indistinguishability of random variables

We consider families of random variables $R = \{ R(x) \}$ where x is a string belonging to some language L which is a particular subset of $\{0,1\}^*$ and all random

variables only take on values in $\{0,1\}^*$. Let $R = \{R(x)\}$ and $S = \{S(x)\}$ be two families of random variables. We select a random sample from either of $R(x)$ and $S(x)$ and give it to a generic "judge" who is to determine from which of $R(x)$ or $S(x)$ our sample comes . We say that $R(x)$ can be replaced by $S(x)$ if for long enough x, as x increases, the verdict of *any* judge becomes irrelevant. Essentially the decision of the judge becomes independent of the family that the sample came from.

Two parameters are immediately apparent in the above scenario : the size of the sample and the amount of time that the judge is allowed to study the sample. By varying the bounds on these parameters we can obtain various degrees of indistinguishability for random variables. We proceed in decreasing order of robustness.

- Equality : if the two families of random variables are exactly alike, then the judge's decision will be meaningless regardless of the sample size and the amount of time that the judge is given to study the sample.
- Statistical indistinguishability : The 2 families of random variables are said to be statistically indistinguishable, if the judge's decision becomes meaningless when given an infinite amount of time to study the sample but the sample's size is polynomially bounded in $|x|$.
- Computational indistinguishability : The 2 families of random variables are said to be computationally indistinguishable if the judge's decision becomes meaningless when he is only given polynomial in $|x|$ time to study the sample and is only given samples of size polynomially bounded from above in $|x|$.

We now proceed to formally define these concepts.

Definition (*Statistical Indistinguishability*). Let $L \subset \{0,1\}^*$ be a language. Two families of random variables $\{ R(x) \}$ and $\{ S(x) \}$ are said to be *statistically indistinguishable* on L if

$$\sum_{\alpha \in \{0\,1\}^*} | \text{prob}\ (\ R(x) = \alpha\ ) - \text{prob}\ (S\ (x) = \alpha\ ) | < |x|^{-c}$$

for all constants $c > 0$ and sufficiently large x.

Before defining computational indistinguishability, it is worth noting that the judge that we will be using will be a polynomial-size family of circuits rather than a polynomial-time Turing machine. The reasons for this will be described after zero-knowledge has been formally defined. Let $C = \{ C_x \}$ be a family of Boolean circuits C_x with outdegree 1 and $|C_x|$ is at most $|x|^e$ for some constant $e > 1$. Our family of random variables $R = \{R(x)\}$, in this case, will be such that all random variables $R(x) \in R$ assign positive probability only to those strings with size exactly $|x|^d$ for some constant $d > 0$ (this to ensure that only samples of polynomial size are presented to the judge). If $C = \{C_x\}$ is a poly-size family of circuits and $U = \{U(x)\}$ is a poly-bounded family of random variables, we denote by $P(R, U, x)$ the probability that C_x outputs a 1 upon input *string* where *string* is a random string distributed according to $U(x)$.

Definition (*Computational or polynomial Indistinguishability*). Let $L \subset \{0,1\}^*$ be a language. Two poly-bounded families of random variables R and S are *computationally or polynomially indistinguishable* on L if for all polynomial-size families of circuits C, for all sufficiently long strings $x \in L$ and for all constants $c > 0$

$$| P(R, C, x) - P(S, C, x) | < |x|^{-c}$$

Note: If R and S are equal, it is obvious that they are also statistically indistinguishable. We can also see that if R and S are statistically indistinguishable then they are also computationally indistinguishable by the following argument. Consider the set I of inputs upon which C_x outputs 1. Since R and S are statistically indistinguishable, $R(x)$ takes on a value in I with almost the same probability that $S(x)$ does. Thus $P(R, C, x)$ will be very close to $P(S,C, x)$.

§2.3.2 Approximating Random Variables

The next logical step is to find an efficient algorithm that will take as input a random variable R and output random strings in a way that is indistinguishable from S. In order to motivate the discussion, recall that we are trying to formalize the notion of the amount of knowledge transmitted by a series of messages. One gains no information from a message which is the result of a feasible (polynomial-time) computation which one could have performed by oneself.

Definition [GMR2]. Let M be a probabilistic Turing machine that halts with probability 1 on input x. For each input x, M will output some string α with a particular probability P_α. We denote by $M(x)$ the random variable that takes on α with probability P_α.

Definition [GMR2]. Let $L \subset \{0,1\}^*$ be a language and $U = \{U(x)\}$ be a family of random variables. We say that U is *perfectly approximable* on L if there exists a probabilistic Turing machine M, running in expected polynomial-time (i.e. the average time for generating $M(x)$ is polynomial in $|x|$), such that $\forall\, x \in L$, $M(x)$ is equal to $U(x)$.

We say that U is *statistically* (*computationally*) *approximable* on L if there exists a probabilistic Turing Machine M, running in expected polynomial time, such that the families of random variables $\{M(x)\}$ and $\{U(x)\}$are statistically (computationally) indistinguishable on L.

§2.3.3 Zero-Knowledge Protocols and Proofs.

As mentioned in the introduction, a protocol should only be considered zero-knowlege if no additional information is leaked during a conversation between a prover and a verifier, even if the verifier is dishonest and attempts to cheat the prover into revealing extra knowledge. So we first consider a cheating verifier who is allowed to deviate arbitrarily from the protocol.

Definition [GMR2] : Let (P,V) be an interactive protocol. Let V^* be an interactive Turing machine that has input x and an extra input tape H, where the length of H is bounded from

above by a polynomial in the length of x. (One can think of H as information about x that V^* already posesses. For instance if we consider the language of graph isomorphism, H could be the colouring of the graphs that V^* posesses. Alternatively, H could be the history of past conversations between P and V^* that V^* is attempting to use in order to extract information from P.) For a round of the protocol on common input x and extra input H, we define the view of V^* to be everything that V^* sees. Namely, let σ (and ρ) be the random strings contained in the random tapes of P (and V^*). Let the computation of P and V^* with these random choices consist of n rounds with the verifier V^* starting first, where p_i (and v_i) are the i^{th} messages of P (and V^*) respectively. Then we say that $(\rho,p_1,v_1,....,p_n,v_n)$ is the view of V^* upon input x and H. We let $View_{P,V^*}(x,H)$ be the variable whose value is this view. For convenience we consider each view to be a string from $\{0,1\}^*$ of length $|x|^c$ for some constant $c > 0$.

Definition [GMR2] : Let $L \subset \{0,1\}^*$ be a language and (P,V) be a protocol. Let V^* be a machine as defined above. We say that (P,V) is *perfectly* (*statistically*) (*computationally*) *zero-knowledge on L* for V^*, if the family of random variables $View_{P,V^*}$ is perfectly (statistically) (computationally) approximable on

$$L^* = \{ (x, H) \mid x \in L \text{ and } |H| = |x|^c \}.$$

We say that (P,V) is perfectly (statistically) (computationally) zero-knowledge on L if it is *perfectly* (*statistically*) (*computationally*) *zero-knowledge on L* for **all** probabilistic polynomial time ITM's V^*.

Note: When a user performs P's role in a zero-knowledge protocol, it is the user's responsibility not to cheat, since there is no guarantee that the protocol is secure if P is replaced by a dishonest P^*. Computational zero-knowledge is the most general of the above concepts and will be simply referred to as zero-knowledge. Thus if (P,V) is zero-knowledge it is not possible for even a dishonest verifier to obtain additional information about members of L in polynomial time, during the course of an interaction with P.

Definition : Let $L \subset \{0,1\}^*$ be a language. We say that (P,V) is a perfectly (statistically) (computationally) *zero-knowledge proof system* for L if it is an interactive proof system for L and a perfectly (statstically) (computationally) zero-knowledge protocol on L.

Notes:

1. It is important to realize that the coin tosses of V are a crucial part of the definition of zero-knowledge for V. We can see this more clearly from the following example. Consider the protocol (P,V) and the language L of all composite integers. Let the input be n whose membership in L we want to establish.

Algorithm:

1. V randomly selects x where $1 < x < n$ and $\gcd(x,n) = 1$.

2. V sends $a \equiv x^2 \bmod n$.

3. P responds with y, a randomly selected square root of a.

If the view consists of only the text of the interaction between P and V (a and y) then the above protocol would be zero-knowledge on L. All our simulator would have to do would be to randomly generate x, compute $a \equiv x^2 \bmod n$ and let $y = x$. The resulting distribution would be identical to that of the text of the interaction between P and V. However, we define the view to also include the coin tosses of V (i.e. the x's). Thus if (x, a, y) is randomly selected in $\text{View}_{P,V}(n)$, then our simulator will only be able to simulate this view by actually factoring n. This is because even if we have the random root y, $\gcd(x + y)$ is not 1 or n with probability at least $\frac{1}{2}$ (there are 4 roots if $n = pq$, where p,q are distinct primes) which does not help us in factoring n [De]. Thus if one cannot factor in probabilistic polynomial time, the above protocol is zero-knowledge.

2. As mentioned earlier, V^* also sees an additional string H. The reason for this was independently discovered by [GMR2], Oren[O], and Tompa and Woll[TW]. H may be thought of in different ways as mentioned in §2.3.3. Hypothetically, H could have been generated through interaction with an infinitely powerful prover. We want to ensure that even a verifier with access to H cannot extract additional knowledge from P. This is why

non-uniformity was introduced in our definition of computational zero-knowledge in the form of distinguishing circuits. The motivation for this was that 2 families of random variables can be computationally distinguished if there are circuits that can discriminate between them whenever certain information is wired in the circuits.

3. The machine M simulating V^*'s view is allowed to use V^* in the *strong sense* as defined next. V^*'s probabilistic nature is modelled by providing it with a random read-only tape. During the course of simulating P, M is allowed to "rewind" V a few steps back in the simulated protocol, resetting V^*'s random read-head to where it had been earlier and proceed with the protocol from that point on. To use the Ali Baba analogy this would be equivalent to the process where the fake Mick Ali cannot come out from the passage dictated since he picked the wrong one. In this case Dan R. had to repeat the current scene. Using the above definitions, we present in the next chapter zero-knowledge protocols for various languages.

Chapter 3
Some Zero-knowledge Protocols

§3.1 Introduction

As promised in the previous chapter we now present zero-knowledge protocols for various languages. More interestingly, we also present interactive proof systems for languages which are not known to be in NP but nevertheless belong to IP.

§3.2 Quadratic Residuosity

We will first need the following definitions:

Let N denote the natural numbers.

$Z_x^* = \{ y \in N \mid 1 \leq y \leq x \text{ and } \gcd(x,y) = 1 \}$ and $x \in N$.

For any $y \in Z_x^*$, we say that y is a quadratic residue mod x if there exists $w \in Z_x^*$ such that $w^2 \equiv y \bmod x$; otherwise y is a quadratic non residue mod x.

We define the quadratic residuosity predicate as

$$Q_x(y) = \begin{cases} 0 \text{ if } y \text{ is a quadratic residue mod } x \\ 1 \text{ otherwise} \end{cases}$$

In the protocol to be presented we consider the language of quadratic residues QR, where

$$QR = \{(x,y) \mid x \in N, x \text{ is not prime, } y \in Z_x^*, \text{ and } Q_x(y) = 0\}.$$

Two interacting parties Peggy and Vic are both given as common input a pair of integers (x,y). Peggy knows that $(x,y) \in QR$ and will attempt to prove to Vic that this is the case without revealing any information to him such as the prime factors of x. If indeed $(x,y) \in QR$ then Vic is convinced by the protocol with a high degree of certainty. On the other hand, if $(x,y) \notin QR$ and Peggy attempts to trick Vic, the latter will detect the subterfuge with the same degree of certainty as above.

Protocol on input (x,y) where $|x| = m$

For $i = 1$ to m

1a. Peggy generates u_i where u_i is a random quadradic residue mod x.

1b. Peggy sends u_i to Vic.

2. Vic sends to Peggy a random bit, $bit_i \in \{0, 1\}$.

3a. Peggy performs the following:

if $bit_i = 0$ then

w_i := a random square root of u_i mod x.

else

w_i := a random square root of $(u_i\, y)$ mod x.

3b. Peggy sends w to Vic.

4. Vic performs the following checks.

if ($bit_i = 0$ and $w_i^2 \bmod x = u_i$) or ($bit_i = 1$ and $w_i^2 \bmod x = u_i\, y$) then

Vic accepts.

else

Vic rejects.

end { For i }

End of Protocol

Claim: The above protocol (P,V) is an interactive proof system for QR.

Proof [GMR2]: Consider our verifier Vic interacting with some arbitrary prover P^*. Let $x>1$ and $y \in Z_x^*$. Consider the case for which y is not a quadratic residue mod x. In this case it is not possible for both u_i and $(u_i\, y)$ to have square roots mod x, since $(u_i\, y)$ will be a quadratic non residue and hence does not have square roots mod x. Since Vic's bits bit_i, are secret (until he reveals them), the probability that Vic will accept the i^{th} round is at most $\frac{1}{2}$ and hence the probability that Vic will accept incorrectly in an m-round protocol is at

most $\frac{1}{2^m}$. We now prove that the above interactive proof system is a perfectly zero-knowledge proof system for QR.

Theorem 3.1. The protocol (P,V) above is a perfectly zero-knowledge proof system for QR.

Proof [GMR2]: Consider our arbitrary (possibly dishonest) polynomial ITM V^* interacting with our prover P. We will first describe V^*'s view during an execution of the protocol and then describe a probabilistic Turing machine M that will produce a simulation of a view with the exact distribution as V^*'s view during a real execution of the protocol. If we can prove that the simulated view is distributed in exactly the same way as the view generated during an actual conversation between the prover (P) and the verifier (V^*), then we will have proved the theorem.

Let $(x,y) \in$ QR be the common input to the pair (P,V^*) such that $|x| = m$ and let H be the extra input to V^* (For more information on what H represents please consult §2.3.3). Let the following random variables denote $View_{P,V^*}((x,y),H)$ (recall from §2.3.3 that this variable represents everything V^* sees in an execution of the protocol).

$R, U_1, BIT_1, W_1, U_2, BIT_2, W_2, ..., U_m, BIT_m, W_m$ where

R $\Leftrightarrow$ The string of random bits generated by V^*

U_i $\Leftrightarrow$ Takes on the value of u_i

BIT_i $\Leftrightarrow$ Takes on V^*'s i^{th} mesage to Peggy

W_i $\Leftrightarrow$ Takes on the value of w_i

We can define the distribution of the view in the following manner: R is assigned a random string r. Let V_i be the random variable consisting of $R, U_1, BIT_1, W_1, U_2, BIT_2, W_2, ..., U_m, BIT_m, W_m$. Assume that for some i, $1 \leq i \leq m$, V_i has been assigned some value v_i. We now describe the process by which values can be given to $U_{i+1}, BIT_{i+1}, W_{i+1}$.

Method

U_{i+1} is assigned a random quadratic residue mod x. If we were dealing with V (whom we know will follow the protocol) then BIT_{i+1} would be assigned the $i+1^{st}$ bit of the random string r. However we are dealing with V^* who is allowed to deviate from the protocol, and as such BIT_{i+1} is assigned the value $bit_{i+1} = f(x, y, H, v_i, u_{i+1})$ where f is some {0, 1}-valued function computable in deterministic polynomial time. If $bit_{i+1} = 0$ then W_{i+1} gets assigned w_{i+1}, a random square root of u_{i+1} mod x. If $bit_{i+1} = 1$ then W_{i+1} gets assigned w_{i+1}, this time a random square root of $u_{i+1}y$ mod x.

Having characterized V^*'s view of the protocol we now show that the protocol is a perfect zero-knowledge proof system by describing the operation of an arbitrary probabilistic Turing machine M which will simulate the above view upon input $(x,y) \in \mathbf{QR}$ and a string H. M runs in expected polynomial time and produces a distribution $M((x,y),H)$ which is identical to $View_{P,V^*}((x,y),H)$.

M's Algorithm

M begins by choosing a random string r of appropriate length. Assume that v_i has already been chosen for some i, $1 \le i \le m$. Then M outputs u_{i+1}, bit_{i+1}, w_{i+1} according to the following algorithm.

Repeat indefinitely

bit_{i+1} := a random member of {0,1}

w_{i+1} := a random member of Z_x^*

if $bit_{i+1} = 0$ then

$u_{i+1} := w_{i+1}^2 \mod x.$

else

$u_{i+1} := (w_{i+1}^2\, y^{-1}) \mod x.$

if $bit_{i+1} = f(x, y, H, v_i, u_{i+1})$ then

M outputs u_{i+1}, bit_{i+1}, w_{i+1} and HALT.

end {repeat}

Notes

1. M chooses a random element of Z_x^* by generating random m-bit messages until one is found that is in Z_x^*. This is clearly expected polynomial time.
2. In the above algorithm y^{-1} refers to the element of Z_x^* that when multiplied by y mod x gives 1.

We now show that M operates in expected polynomial time in m and also produces the desired output distribution. Let R', $\{U_i', BIT_i', W_i'\}$ be the random variables corresponding to the output of M, and let V_i' be defined similary to V_i. R' is obviously distributed similarly to R. Let $1 \leq i \leq m$ and assume that V_i' has exactly the same distribution as V_i and assume that M assigns a value to V_i' in time polynomial in m. Say that both V_i and V_i' have been given the value v_i. We want to show that M's code halts in expected polynomial time in m and has the same output distribution as the view characteristic as described in the method above, given that $V_i = V_i' = v_i$.

Consider the body of the repeat loop up to but not including the last test. Every quadratic residue in Z_x^* has the same number of square roots [De]. Thus if $bit_{i+1} = 0$, u_{i+1} is a random quadratic residue and w_{i+1} is a random square root of u_{i+1}. If $bit_{i+1} = 1$, u_{i+1} is also a random residue and w_{i+1} is a random square root of $u_{i+1}y$. This means that the body of the loop is equivalent to the following (although what follows may not be efficiently executable):

u_{i+1} := a random quadratic residue mod x.

bit_{i+1} := a random element of {0 1}.

if $bit_{i+1} = f(x, y, H, v_i, u_{i+1})$ then

    if $bit_{i+1} = 0$ then

        $w_{i+1} := \sqrt{u_{i+1}} \bmod x$.

    else

        $w_{i+1} := \sqrt{u_{i+1}\, y} \bmod x$.

HALT and output $(u_{i+1}, bit_{i+1}, w_{i+1})$

end if.

The above equivalent code halts and outputs with probability $\frac{1}{2}$ and hence the actual repeat loop halts in expected polynomial time in the length of the input. The equivalent code is as likely to halt for any value of u_{i+1} and hence U'_{i+1} gets assigned a random quadratic residue. BIT'_{i+1} will be assigned the value $f(x, y, H, v_i, u_{i+1})$. When the code does halt, W'_{i+1} gets assigned a random square root of u_{i+1} y mod x or u_{i+1} mod x depending on bit_{i+1} as required.❑

§3.3 Quadratic Non Residuosity

The protocol presented in [GMR2] is a simplified version of the protocol put forward in [GMR1] and makes use of cryptographic capsules invented by Cohen [Co]. For a discussion of cryptographic capsules refer to Appendix A. In the protocol to be presented we consider the language of quadratic non residues, **QNR**, where

$$\mathbf{QNR} = \{(x,y) \mid x \in N, x \text{ is not prime}, y \in Z_x^*, \text{ and } Q_x(y) = 1\}$$

As previously, two interacting parties Peggy and Vic are both given as common input a pair of integers (x,y). Peggy knows that $(x,y) \in \mathbf{QNR}$ and will attempt to prove to Vic that this is the case without revealing any information to him such as the prime factors of x. If indeed $(x,y) \in \mathbf{QNR}$ then Vic is convinced by the protocol with a high degree of certainty. On the other hand, if $(x,y) \notin \mathbf{QNR}$ and Peggy attempts to trick Vic, the latter will detect the subterfuge with the same degree of certainty as above. Let (P,V) be an interactive protocol given as input (x,y) where $|x| = m$. We first give an informal description of the protocol. Informally, Vic generates random elements of the following 2 types:

1. $w = r^2 \bmod x.$ $\quad r \in Z_x^*$
2. $w = yr^2 \bmod x.$ $\quad r \in Z_x^*$

Vic flips a coin to determine which of the above types to send to Peggy in any round. If $(x,y) \in$ **QNR**, then Peggy can easily determine which of type 1 (w is a quadratic residue mod x) or 2 (w is a quadratic non residue mod x) Vic sent. If $(x,y) \notin$ **QNR**, then w is always a quadratic residue regardless of whether it is of type 1 or 2 and Peggy's chance of guessing the type of w is no better than $\frac{1}{2}$. By repeating this protocol m times Vic's chance of being fooled by a cheating Peggy becomes exponentially small. Thus, the above qualifies as an interactive proof system but it is *not* a zero-knowledge proof system. This is because if we are dealing with an arbitrary verifier, the latter may have deviated from the protocol in generating the w's in order to extract additional information from Peggy. Such knowledge could consist of finding out whether a particular w is a quadratic residue or not mod x. In order to avoid this we further require that the verifier prove to Peggy that the w's were generated as specified by the protocol. Essentially the verifier has to prove to Peggy that he knows the type of w without revealing the type. This can be done by using the method of cryptographic capsules and residue classes as described in [Co]. We now describe an interactive protocol which is a statistical zero-knowledge proof system for **QNR**. This protocol should be repeated m times upon input (x,y).

Protocol on input (x,y) where $|x| = m$

For $i = 1$ to m

1a. Vic chooses a random $r \in Z_x^*$ and $bit \in \{0,1\}$ at random.

if $bit = 0$ then

Vic sets $w := r^2 \bmod x$.

else

Vic sets $w := yr^2 \bmod x$.

1b. Vic sends w to Peggy.

1c. In this part of the protocol Vic now has to prove to Peggy that he indeed has followed the protocol and knows the type of w without revealing it to

Peggy. We now give an example of an implementation of cryptographic capsules as discussed in Appendix A.

for $j = 1$ to m

Vic chooses r_{j1} and $r_{j2} \in Z_x^*$ and $bit_j \in \{0\ 1\}$ at random.

Vic sets $a_j := r_{j1}^2 \bmod x$ and $b_j := yr_{j2}^2 \bmod x$.

if $bit_j = 1$ then

Vic sends to Peggy the ordered pair, $pair_j = (a_j, b_j)$

else

Vic sends to Peggy the ordered pair, $pair_j = (b_j, a_j)$

end {for }

In the language of cryptographic capsules $pair_j$ becomes our capsule

2. Peggy sends to Vic an m-bit string $i = i_1 i_2 i_3 \ldots i_m$.

3. Vic sends to Peggy the sequence $v = v_1\ v_2\ v_3 \ldots\ v_m$ constructed as follows

If $i_j = 0$ then

(Vic opens the capsule)

$v_j = (r_{j1}, r_{j2})$

else

(Vic reveals he knows how w was constructed)

if $bit = 0$ then

$v_j := rr_{j1} \bmod x$ (a square root of $wa_j \bmod x$)

else

$v_j := yrr_{j1} \bmod x$ (a square root of $wb_j \bmod x$)

4a. Peggy now verifies whether the sequence v was constructed properly

for $j = 1$ to m, Peggy performs the following checks

if $i_j = 0$ then

$v_j = (s,t)$ where $(s^2 \bmod x,\ t^2 y \bmod x) = pair_j$

else

$(v_j^2)w^{-1} \bmod x \in pair_j$

If any of the above 2 conditions do not hold then Peggy halts the protocol and rejects, else

4b. If $w \in$ **QR** then

Peggy sets *answer* := 0

else

Peggy sets *answer* := 1

4c. Peggy sends *answer* to Vic

5 Vic does the following checks

if *answer* = *bit* then

Vic continues the protocol

else

Vic rejects and halts

end { For }

End of Protocol

If Vic has not rejected after m iterations of the protocol, Vic accepts and halts.

Note : Step 3 can be thought of intuitively as follows: if $i_j = 0$ then Vic is convincing Peggy that $pair_j$ was constructed properly. If $i_j = 1$ then Vic is convincing Peggy that, if $pair_j$ was chosen properly, w was constructed as specified in the protocol. Vic is only able to do that if he actually constructed w in the manner specified by the protocol. If he were to cheat and try to use a specific value for w he would only be able to answer Peggy's queries by finding the square root of w, which would involve factoring x. However the factors of x consist of Peggy's secret and Vic is unable to factor x in polyomial (in $|x|$) time.(If Vic

knew the factors of x he would not need Peggy to tell him whether w is a quadratic residue or not)

Claim: The above protocol (P,V) is an interactive proof system for QNR

Proof: (P,V) is clearly an interactive protocol. If $(x,y) \in$ QNR then w is a quadratic non residue iff *bit* =1 and P can always determine whether w is a quadratic residue or not. Thus P can always calculate *answer* such that V will always accept. On the other hand, if $(x,y) \notin$ QNR and V is interacting with a dishonest prover P^*, then even if P^* has infinite power she cannot distinguish the case where *bit* = 0 from the case where *bit* = 1. This is because the messages P receives from V, consisting of w and $pair_j$, will consist only of *random* quadratic residues and this yields absolutely no information on the value of *bit*. Now consider step 3 of the protocol where P^* receives the pair (r_{j1}, r_{j2}) for the case $i_j = 0$. Note that r_{j1} is a random square root of a_j and r_{j2} is a random square root of $b_j y^{-1} \bmod x$. These too give no information on the value of *bit*. Now consider the items received by P^* in step 3 when $i_j = 1$. If *bit* = 0 then P^* receives rr_{j1} which is a random square root of $wa_j \bmod x$. If *bit* = 1 then P^* receives rr_{j2}, a random square root of $wb_j \bmod x$. Since $pair_j$ is a random ordering of (a_j, b_j), v_j is equally as likely to be a random square root of w times the first element of $pair_j$, as it is to be a random square root of w times the second element of $pair_j$ regardless of the value of *bit*. Thus from the information received by P^*, the value of *bit* is as likely to be 0 or 1 and the probability that P^* correctly guesses the value of *bit* is no better than $\frac{1}{2}$. Given that the protocol consists of m stages, the probability that P^* correctly guesses the value of *bit* for all m iterations is at most 2^{-m}. We next show that (P,V) is statistically zero-knowledge for QNR. This is quite complex but we give a full proof as it illustrates quite well the definitions and properties given in chapter 2.

Theorem 3.2: The above protocol is a statistically zero-knowledge proof system for QNR.

Proof [GMR2]: Consider our arbitrary probabilistic polynomial-time interactive Turing

machine V^* interacting with our prover P. As before we first describe V^*'s view during an execution of the protocol and then describe a probabilistic Turing machine M that will produce a simulation of a view with the exact distribution as V^*'s view during a real execution of the protocol. If we can prove that the simulated view is distributed in exactly the same way as the view generated during an actual conversation between the prover (P) and the verifier (V^*), then we will have proved the theorem.

Let $(x,y) \in$ QNR be the common input and $|x| = m$ and let H be the extra input to V^*. For convenience, consider the random variable $View_{P,V^*}((x,y),H)$, representing V^*'s view of an iteration of the protocol, to consist of the following random variables:

RAN, and

$\{W^k, \{PAIR_j^k : 1 \le j \le m\}, \{I_k^j : 1 \le j \le m\}, V^k, ANSWER^k \mid 1 \le k \le m\}$

where

RAN $\Leftrightarrow$ String of random bits generated by V^*.

W^k $\Leftrightarrow$ Value of w in the kth round of the protocol.

$PAIR_j^k$ $\Leftrightarrow$ Value of $pair_j$ in the k^{th} round of the protocol.

I_k^j $\Leftrightarrow$ Value of i_j in the k^{th}round of the protocol.

V^k $\Leftrightarrow$ Value of the sequence v in the k^{th} round of the protocol.

$ANSWER^k$ $\Leftrightarrow$ Value of *answer* in the k^{th} round of the protocol.

For explanation purposes and clarity, we will concentrate on showing that a single iteration of the protocol is statistically zero-knowledge. The proof can be generalised in the same manner as was done in the proof of the previous theorem and involves carrying the view of the protocol up to the current point. We therefore drop all superscripts and consider only the following random variables:

RAN, W, $\{PAIR_j\}$, $\{I_j\}$, $V=V_j$, ANSWER.

It should be noted that during an *honest* execution of the protocol, we expect that

$$W = \begin{cases} r^2 \bmod x \text{ or} \\ r^2 y \bmod x \end{cases} \qquad r \text{ is a substring of RAN}$$

and

$$PAIR_j = \begin{cases} (r_{j1}^2 \bmod x,\ y r_{j2}^2 \bmod x) \text{ or} \\ (y r_{j2}^2 \bmod x,\ r_{j1}^2 \bmod x) \end{cases} \qquad r_{j1}, r_{j2} \text{ are substrings of RAN}$$

and

If $I_j = 0$ then

$$V_j = (r_{j1}, r_{j2})$$

else

$$V_j = \begin{cases} r r_j \bmod x \text{ or} \\ y r r_j \bmod x \end{cases} \qquad \text{depending on the value of } bit.$$

However, V^* may not follow the protocol and hence, the only thing we can say about these random variables is that:

RAN is assigned a random binary string,

W and $PAIR_j$ are assigned values w and $pair_j$ computed by V^* upon inputs x, y, H and RAN,

I is a random binary string of length m, and

V_j is computed by V^* upon inputs x, y, H, RAN and I.

We can now describe a probabilistic Turing machine M, which runs in expected polynomial time and will output a distribution that is statistically indistinguishable from $View_{P,V^*}((x,y),H)$, given $(x,y) \in QNR$ and H.

M's algorithm

1. M outputs a random string *ran* of the appropriate length and runs V^* on inputs (x,y,H) and random tape *ran*. V^* goes through step 1 producing w, $pair_j$, for $1 \le j \le m$.

2. M next picks random $i_1,...i_m$ where $i_j \in \{0,1\}$, sets $i = i_1 i_2 i_3 ...i_m$ and writes i onto V^*'s communication tape thus activating step 3. In effect, M is playing Peggy's role.

3. V^* then goes through stage 3, outputting a sequence $v = \{v_j\}$. M outputs w, $\{pair_j\}$, i, and v. M then performs the checking P would have done (step 4a). If the checks fails M outputs "terminate" and halts.

Let us assume that the checks succeeded. We can now think of x, y, *ran*, and H as being fixed so that w and $\{pair_j\}$ are also fixed. Since the checks succeeded, it implies that P would at this point be ready to send to V^* the value of *answer*. Let us call those i 's which result in such cases *special*. Therefore M has discovered that i is special and now has to compute the value of *answer* that P would have sent. This value is 0 if $w \in \mathbf{QR}$ and 1 otherwise. Since V^* might not have computed w in the same way that V would have and M does not know the factorization of x, it is not immediately obvious as to how *answer* can be calculated. We will now show that this can be done if we can find another special string i' such that $i' \neq i$.

Suppose that $i_j = 0$ and $i'_j = 1$ where i and i' are special. Let v, v' be the sequences sent by V^* after receiving i and i' respectively. These can be constructed in polynomial time by running V^*. Since $i_j = 0$, $v_j = (s,t)$ where $(s^2 \bmod x, t^2 y \bmod x)$ is equivalent to $pair_j$ with the elements possibly interchanged. Also since $(i'_j = 1, (v'_j)^2 w^{-1}) \bmod x \in pair_j$. It follows that if $(v'_j)^2 w^{-1} = s^2 \bmod x$, then w is a quadratic residue mod x and if $(v'_j)^2 w^{-1} = t^2 y \bmod x$, then w is a quadratic non residue mod x. It now remains to find a special $i' \neq i$. The following algorithm does this.

Algorithm for finding special $i' \neq i$.

2^m random strings i' are tested (with replacement) until either a special $i' \neq i$ is found or 2^m strings have been tested. In the event we do not find such a special i we then test *all* 2^m strings looking for that special i.

If a special $i' \neq i$ is found then M calculates the value of *answer* as explained above. If no such i exists, M outputs "?" which will happen when i is the *only* special string. We now have to show that this algorithm operates in expected polynomial time. We show this for each fixed value of x, y, H, and *ran*. Let x, y, H, and *ran* be fixed which implies that w and *pair* are also fixed. Let k be the number of strings that are special.

Case 1: $k = 0$

In this case, the algorithm above will not even be invoked and the running time of M is clearly polynomial in m.

Case 2: $k = 1$

In this case, the probability that M will choose a special i is $\frac{1}{2^m}$ and our algorithm runs for time $2^m m^c$ (for some c) and the expected running time is $\left\lceil \frac{2^m m^c}{2^m} + \text{a polynomial in } m \right\rceil$.

Case 3: $k > 1$

In this case, the probability that M will choose a special i is $\frac{k}{2^m}$. We want to calculate an upper bound on the expected running time of our algorithm and we do this by stipulating that the algorithm will only stop when a special string $i' \neq i$ is found. In effect, we are tossing a biased coin where the probability of "heads" is $\frac{k}{2^m}$. This is a geometric distribution and the expected number of coin tosses required to obtain the first head is exactly $\frac{2^m}{k-1}$. Hence, the expected running time for our algorithm is $\leq \left\lceil \frac{2^m}{k-1} \right\rceil \cdot m^c$ (for some constant c). Thus the total expected time is $\leq \left\lceil \frac{k \cdot 2^m \cdot m^c}{2^m \cdot (k-1)} + \text{a polynomial in } m \right\rceil$ which is also a polynomial in m.

Recall that M((x,y),H) is the random variable denoting the distribution of M's output given x, y and H. It remains to show that M's view is statistically close to $\text{View}_{P,V^*}((x,y),H)$.

Fix x, y and H. If *ran* is such that the number of special strings is not exactly 1, any output string α beginning with *ran* is taken on with exactly equal probability by both $M((x,y),H)$ and $View_{P,V^*}((x,y),H)$. This is because we will always be able to find a special string and hence calculate the value of answer. Let S be the set $\alpha = \{ran, w, \{pair_j\}, i, v, answer\}$ where i is the *unique* special string determined by *ran*. The probability that $View_{P,V^*}((x,y),H)$ takes on a value in S is $\leq \frac{1}{2^m}$ since for each *ran* there is at most one i which will be the unique special string. Similarly, the probability that $M((x,y),H)$ takes on a value in S is $\leq \frac{1}{2^m}$. Thus

$$\sum_{\alpha} |Prob(\ M((x,y),H) = \alpha\ ) - Prob(\ View_{P,V^*}((x,y),H) = \alpha)\ | \quad =$$

$$\sum_{\alpha \notin S} |Prob(\ M((x,y),H) = \alpha\ ) - Prob(\ View_{P,V^*}((x,y),H) = \alpha)\ | \quad +$$

$$\sum_{\alpha \in S} |Prob(\ M((x,y),H) = \alpha\ ) - Prob(\ View_{P,V^*}((x,y),H) = \alpha)\ | \quad \leq$$

$$0 + \frac{1}{2^m} + \frac{1}{2^m} = \frac{2}{2^m}.$$

Note: The reason we have two terms for the case where $\alpha \in S$ is that the value of *ran* that would give a unique i can be chosen by either M or by V^* during a real conversation with the prover. Both these cases will result in failure for our simulating machine.

Since the protocol consists of m rounds, the difference is

$$m \sum_{\alpha} |Prob(\ M((x,y),H) = \alpha\ ) - Prob(\ View_{P,V^*}((x,y),H) = \alpha)\ | \leq \frac{2m}{2^m}$$

This completes our proof.❑

§3.4 Graph Isomorphism [GMW1]

This problem can be stated as follows. Peggy and Vic have as common input two graphs G_1 and G_2. Peggy knows that the two graphs are isomorphic and will prove this to Vic without revealing the isomorphism or any additional information. This proof is presented here for two reasons. First, the fact that a zero-knowledge proof exists without revealing the isomorphism is in itself interesting. Second, our simulating probabilistic polynomial time machine M will make use of the "rewinding" process discussed in Chapter 2. In case of failure M will rewind itself to the last point in the simulation where it was successful and try again with new random choices. In effect, cases resulting in failure are ignored. In the protocol due to Goldreich, Micali, and Widgerson [GMW1] the prover only needs to be a probabilistic polynomial time machine which obtains as auxilliary input the isomorphism between the input graphs. Constrast this with our earlier model in which the prover was infinitely powerful.

Let $G_1(V,E_1)$ and $G_2(V,E_2)$ be the two graphs entered as common input to our protocol where V is the set of vertices and E_i refers to the edges in the graph G_i. Let ϕ be the isomorphism between the two graphs G_1 and G_2. In the following protocol Peggy proves to Vic that the two graphs are isomorphic without revealing the isomorphism. The following steps should be repeated n times where $n = |V|$, using random coin tosses.

Repeat n times

1a. Peggy generates H, a random isomorphic copy of G_1. This can be done by selecting a random permutation $\pi \in sym(V)$, where $sym(V)$ is the symmetric group of all permutations of V, and computing H(V,F) such that $(\pi(u),\pi(v)) \in F$ iff $(u,v) \in E_1$.

1b. Peggy sends the graph H(V,F) to Vic.

2. Vic chooses a random $\alpha \in \{1, 2\}$ and sends α to the prover. In effect Vic is asking that Peggy demonstrate that H and G_α are indeed isomorphic.

3. If $\alpha \notin \{1, 2\}$ (which means that Vic is obviously cheating) Peggy rejects and halts. If $\alpha = 1$, Peggy sends π to the verifier, else she sends $\pi\phi^{-1}$.

4. If the permutation received from Peggy is not an isomophism between H and G_α, Vic stops and rejects. Otherwise he continues the protocol.

If *n* successful iterations of the protocol have been completed by Vic, he then accepts.

Analysis

The above protocol can be easily shown to be an interactive proof system for graph isomorphism. If Peggy really knows the isomorphism, she can respond to either challenge. If she doesn't know it and tries to cheat she can only respond to one of the challenges and will be caught with probability $\frac{1}{2}$ during each round. After *n* rounds the probability that Vic would accept incorrectly is at most $\frac{1}{2^n}$. This probability can be made exponentially small by increasing the value of *n*.

Intuitively, this protocol is zero-knowledge since all the verifier receives from the prover are random isomorphic copies of the common input which she could obviously have computed on her own. In any round of the protocol the verifier obtains either π, a random permutation which reveals no information about the value of ϕ, or the composition $\pi\phi^{-1}$ also being a random permutation which reveals nothing about ϕ. We now give the more formal proof of zero-knowledge showing that no knowledge is revealed to the specified verifer but also to *any* verifier including those which are allowed to deviate arbitrarily from the protocol.

Theorem 3.3: The above protocol constitutes a zero-knowledge interactive proof system for Graph Isomorphism.

Proof[GMW1]: As before let V^* be an arbitrary probabilistic polynomial-time Turing machine interacting with the prover. V^* is allowed to deviate arbitrarily from the protocol. We now describe a probabilistic polynomial time machine M that generates a probability distribution which is identical to that induced on V^*'s tapes during an actual interaction with the prover. We demonstrate its existence by construction. We proceed by trying to guess which isomorphism V^* will ask to check. Thus, we will construct some graph H such that we will be able to answer in case we guessed right. The cases where we guessed wrong will be ignored. It is essential from V^*'s point of view that the cases which lead to failure and those which lead to success look identical. In this way, throwing away those instances in which we failed merely slows down the construction without actually affecting the probability distribution that V^* sees. We will now describe M's operation when given as input the graphs G_1 and G_2. M will monitor V^*'s execution and will enact the role of the prover to V^*. M begins by chosing a random string *ran* of the appropriate length and placing it on V^*'s random tape and its own record tape. In effect this fixes V^*'s random coin tosses. M proceeds as follows.

M's algorithm

1. M picks random $\beta \in \{1, 2\}$ and a random permutation $\pi \in \mathrm{Sym}(V)$. It then computes H(V,F) such that $(\pi(u),\pi(v)) \in F$ iff $(u,v) \in E_\beta$. M then places H on V^*'s communication tape.
2. M now reads V^*'s request from its communication tape. If the request is $\alpha = \beta$ (lucky for M), M appends (H,α,π) to its record tape and proceeds to the next round.

If $\alpha \notin \{1, 2\}$ in which case V^* is obviously cheating, M appends (H,α) to its record and stops. If $\alpha + \beta = 3$ (unlucky for M), M must repeat the current round. This is done by first rewinding V^* to its configuration at the beginning of the current round and repeating steps 1 and 2 above with new random choices. If all rounds are successfully completed M

outputs its record tape and halts. It should be noted that Prob($\beta = 1 | H^{(i)}$) $= \frac{1}{2}$ where $H^{(i)}$ is the list of graphs sent so far (including those sent in the current repetition of round i, but excluding those sent after V^* was rewound). This says that the value of β is independent of the graphs sent so far. Thus Prob($\beta = \alpha(ran,H^{(i)}) | H^{(i)}$) $= \frac{1}{2}$, where $\alpha(ran,H^{(i)})$ is V^*'s answer upon random tape *ran* and communication tape $H^{(i)}$. Thus the probability that the i^{th} round is repeated j times is at most 2^{-j}. M stops and outputs its record after n rounds have been completed or after an invalid $\alpha \notin \{1\ 2\}$ has been encountered. If n rounds were successfully completed M outputs a triple of the form (H,α,π) where π is an isomorphism between H and G_α. It is easy to see that in both cases M outputs the right distribution. Furthermore the probability distribution output by our simulator is *identical* to that observed during an actual interaction between V^* and the real prover. Thus our protocol is a perfectly zero-knowledge proof system for Graph Isomorphism.❑

§3.5 Graph Non Isomorphism

We now present an interactive proof system for graph non isomorphism due to Goldreich, Micali, and Widgerson [GMW1]. This is particularly interesting since, although it has been shown that zero-knowledge proofs exist for all statements in NP (See Chapter 4), graph non isomorphism is not known to be in NP. This has significant ramifications for the size of the class IP. In the following protocol the prover only needs to be a probabilistic polynomial time machine with access to an oracle for graph isomorphism.

Let the common input to our protocol (P,V) be the graphs $G_1(V,E_1)$ and $G_2(V,E_2)$. The following constitutes a 2-move interactive protocol.

1. The verifier (Vic) chooses n random integers $\alpha_i \in \{1, 2\}$, $1 \le i \le n$. He then computes n random graphs $H_i(V,F_i)$ such that F_i is a random isomorphic copy of G_{α_i}. Vic sends all the H_i's to the prover (Peggy).

2. Peggy replies with a string of β_i's $\in \{1, 2\}$ such that $H_i(V,F_i)$ is isomorphic to $G_{\beta_i}(V,E_{\beta_i})$.

3. Vic now checks whether $\alpha_i = \beta_i \; \forall \; i$, $1 \leq i \leq n$. If the condition does not hold for at least one i, Vic rejects else he accepts.

Theorem 3.4: The above protocol constitutes a two-move interactive proof system for graph non isomorphism.

Proof: If G_1 and G_2 are not isomorphic to each other and both parties follow the protocol, Peggy can always distinguish which of G_1 or G_2 the H_i's are isomorphic to and Vic will always accept. If G_1 and G_2 are isomorphic to each other and Peggy attempts to cheat, then, due to the randomness of the permutation, the H_i's are as likely to be isomorphic to either graph. Since Peggy does not see the α_i 's, the probability that she replies with the right β's is at most $\frac{1}{2}$ for each i and hence the probability that Vic accepts incorrectly is at most 2^{-n}. This can be made arbitrarily small by increasing the value of n. The above protocol is not zero-knowledge as a cheating verifier could use the prover to determine to which of the input graphs a particular graph G_3 is isomorphic. This problem can be remedied by first requiring the verifier to prove to the prover that he knows an isomorphism between the query graph H_i and one of the input graphs. This is done by using the principle of cryptographic capsules in a similar way that was used in the Quadratic Non Residuosity protocol of §3.3 where the verifier was required to prove that he knew the type of w. In the following case, residue classes are replaced by equivalence classes induced by graph isomorphism and class equivalence is demonstrated by exhibiting permutations.❑

§3.6 Graph 3-Colourability

This protocol due to Goldreich, Micali, and Widgerson [GMW2] is presented as it is will be used in Chapter 4 in the proof that all statements in NP have zero-knowledge proof

systems. The existence of secure encryption schemes is assumed (as described by Goldwasser and Micali [GM]). An encryption scheme secure as in [GM] is an algorithm f, that upon input x and internal coin tosses r, outputs an encryption $f(x,r)$ where f should obey the following properties:

1. f should be computable in polynomial time.
2. Decryption should be unique. That is

   $$f(x,r) = f(y,s) \Rightarrow x = y$$

3. It is impossible to distinguish any encryption $f(x,r)$ from a random string in polynomial time.

The graph 3-colourability problem can be summarized as follows: Given a graph G, can the vertices of G be coloured using 3 colours such that adjacent vertices are coloured differently? Let G(V,E) be the common input to our protocol. Note that in our protocol the prover only needs to be a probabilistic polynomial-time machine with access to an oracle for a proper 3-colouring of our input graph. We denote the colouring as ϕ such that $(\phi:V\rightarrow\{1, 2, 3\})$. Let the number of vertices $n = |V|$ and the number of edges $m = |E|$. For convenience, we number $V=\{1\ 2....\ n\}$.

Protocol

Repeat the following steps m^2 times

1a. The prover, Peggy, generates a random permutation π of the 3-colouring and random strings $r_1,r_2.....r_n$.

1b. Peggy computes $R_i = f(\pi(\phi(v)),r_i)$ for every $v \in V$.

1c. Peggy sends the R_i's to the verifier Vic.

2. Vic picks a random edge $e \in E$ and sends it to Peggy.

3. If $e = (u,v) \in E$, Peggy reveals the colouring of u and v as per step 1b and "proves" that they correspond to their encryption. More precisely, Peggy sends $(\pi(\phi(u)),r_u)$ and $(\pi(\phi(v)),r_v)$ to Vic.

   If $e \notin E$, Peggy halts and rejects.

4. Vic now verifies Peggy's proof which has to meet the following 4 conditions

   i. $R_u = f(\pi(\phi(u)), r_u)$

   ii. $R_v = f(\pi(\phi(v)), r_v)$

   iii. $\pi(\phi(u)) \neq \pi(\phi(v))$

   iv. $\pi(\phi(u))$ and $\pi(\phi(v)) \in \{1, 2, 3\}$

   If any of these above conditions is violated, Vic rejects. Otherwise he goes on to the next round of the protocol.

Claim:The above protocol constitues an interactive proof system for graph 3-colourability.

Proof: If the graph is 3-colourable and both Peggy and Vic follow the protocol, the latter will always accept. Consider now the case of a cheating prover. If Peggy does not know the colouring or no such colouring exists, then there exists at least one edge $(u,v) \in E$ such that $\phi(u) = \phi(v)$ and hence $\pi(\phi(u)) = \pi(\phi(v))$. Thus, in any round, Peggy will be caught cheating with probability at least $\frac{1}{m}$. After m^2 rounds, the probability that Peggy will have been caught is $\geq \left[\frac{1}{m}\right]^{m^2} \geq \left[\frac{1}{e}\right]^m$ as required.

Theorem 3.5: If $f(;\ ,\ ;)$ is a secure probabilistic encryption, then the above protocol constitutes a zero-knowledge interactive proof system for graph 3-colourability.

Proof (Sketch): It is clear that the prover reveals no additional knowledge to the specified verifier. However, we require that no knowledge be leaked to *any* verifier, including dishonest ones. As before, consider V^*, an arbitrary verifier interacting with our prover P. We will describe a probabilistic polynomial-time machine M that will generate a probability distribution which is polynomially indistinguishable from the distribution induced on V^*'s tapes during its interaction with P. The construction of M follows.

M will monitor the execution of V^* by fixing its random tape and reading from and writing to its communication tapes. Basically, M attempts to guess the edge that V^* will ask to check and encrypts an illegal colouring of G such that it can answer in case it guessed right. Those attempts resulting in failure are ignored and here again M simply "rewinds" V^* to its

configuration after its last success and tries again with new random choices. Again, from V^*'s point of view, it is crucial that those cases that lead to success be polynomially indistinguishable from those resulting in failure. M begins by picking a random tape *ran* and writes it on V^*'s random tape and its own record tape.

M's Algorithm

Repeat m^2 times

1a. M picks a random edge $(u,v) \in E$ and random integers $(a,b) \in \{(i,j): 1 \leq i \neq j \leq 3\}$. It then picks random r_i's and computes R_i as follows:

$$R_i = f(c_i,r_i) \quad \text{where } c_i = 0 \quad \text{for } i \in V - \{u,v\},\ c_u = a,\ c_v = b.$$

1b. M places the R_i's on V^*'s communication tapes.

2. M now reads V^*'s request edge e. If $e \notin E$, V^* is obviously cheating and M appends the R_i's and e to its record tape and stops.

If $e \neq (u,v)$ (unlucky for M), M rewinds V^* to its configuration at the beginning of the current round and repeat the round using new random choices. If $e = (u,v)$ (lucky for M), M places (a,r_u) and (b,r_v) on V^*'s communication tape. Finally, it places the R_i's, e, (a,r_u), (b,r_v) to its record tape and goes on to the next round.

If all m^2 rounds are successfullly completed, M outputs its record tape. In a technical lemma (to appear in a final version of [GMW2]), the authors prove that the three possible replies of the verifier, namely $e \notin E$, $e \in E - (u,v)$ and $e = (u,v)$, occur with essentially the same probability during the simulation as they do during an interaction between V^* and the real prover. Thus the probability that a particular round will have to be repeated more than km times is smaller than 2^{-k} so that M completes in polynomial time. The only difference between the probability distribution generated during a real interaction and that generated during our simulation, is that the former contains probabilistic encryptions of colourings whereas the latter consists of encryption of mostly 0's. However, by property (3) of our

encryption function, we find that both these types are indistinguishable in polynomial time.❑

§3.7 Directed Hamiltonian Cycle [Bl2]

This protocol is due to Blum and the problem can be described as follows: Given a Hamiltonian graph G with n vertices as common input the following protocol allows Peggy to convince Vic that she knows a Hamiltonian cycle H in G without revealing any information about H. Let $f(x,r)$ be an arbitrary encryption scheme as per §3.6.

Repeat n times

1a. Peggy generates a random permutation π of G and permutes the vertices of G to yield $G' = \pi(G)$ and generates n^2 random strings r_{ij} where $1 \leq i,j \leq n$. She then constructs the adjacency matrix A of G' where

$$A = [a_{ij}]$$

1b. Peggy computes $R_{ij} = f(a_{ij},r_{ij})$ for all i,j and sends them to Vic. At this point Peggy is committed to the values of the adjacency matrix.

2. Vic picks a random bit b and sends it to Peggy.

3. If $b = 0$ Peggy sends π and all the r_{ij} 's to Vic who checks that the R_{ij};s indeed encrypt the adjacency matrix of G'. If $b = 1$ Peggy sends n values of R_{ij} such that the edges (i,j) form a valid directed cycle in G'. Vic checks that these correspond to a cycle in G' from the structure of the adjacency matrix. If any of these conditions does not hold Vic rejects. Otherwise he goes on to the next round of the protocol.

Theorem 3.6: The above protocol constitutes an interactive proof system for the DHC.

Proof: If Peggy knows a cycle H and both parties follow the protocol Vic will always accept. Consider the case where Peggy does not know H and attempts to cheat. She can either send encryptions of the valid adjacency matrix or she can send fake encryptions. In the first case she cannot answer the challenge (b =1) and in the latter case she cannot

answer (b =0). Thus she is caught cheating with probability $\frac{1}{2}$ in each round. After n rounds the probability that Vic is fooled is less that 2^{-n}.

Theorem 3.7: If f is a secure encryption scheme, the above protocol is a zero-knowledge interactive proof system for the DHC.

Proof: The construction of the simulating machine is similar to the one in the previous section and is left as an exercise for the reader.

§3.8 The Discrete Log Problem

The protocol that will be presented allows Peggy to prove to Vic that she knows the solution to the Discrete Log Problem (for more on this problem see [Od]). This means that Peggy demonstrates she knows x such that $\alpha^x \equiv \beta \bmod N$ but does not reveal the value of x or any extra knowledge to Vic. The protocol we present was first put forward in a somewhat more complex form by Chaum, Evertse, Van de Graaf and Peralta [CEPG] and subsequently refined and improved by Chaum, Evertse, and Van de Graaf [CEG] and it is the latter version that we will describe.

Given N, $\alpha \in Z_N^*$, $\beta \in <\alpha>$, where $<\alpha>$ is the group generated by α, demonstrate that we know x such that $\alpha^x \equiv \beta \bmod N$, where N is either a prime or the product of two primes,where the primes are of order $O(\sqrt{N})$. If N is composite, it is assumed that the prover knows its factorization. In the following protocol, Peggy proves to Vic that she knows x such that $\alpha^x \equiv \beta \bmod N$.

Repeat the following T times

1a. Peggy chooses random $r \in \{1,...,\phi(N)\}$.

1b. She then computes $\gamma \equiv \alpha^r \bmod N$ and sends γ to Vic.

2. Vic chooses random $b \in \{0\ 1\}$ and sends b to Peggy.

3. Peggy computes $y \equiv r + bx \bmod \phi(N)$ and sends y to Vic.

4. Vic checks that $\alpha^y \equiv \gamma\beta^b \bmod N$.

If condition 4 is violated in any round, Vic stops and rejects. Otherwise he accepts. In the above protocol ϕ refers to Euler's totient function. For an integer n, $\phi(n)$ is defined as the number of elements of the set $\{0,...n\text{-}1\}$ that are relatively prime to n [De].

Theorem. The above protocol is an interactive proof system for Discrete Log.

Proof: If Peggy does not know x, the she will not be able to repond with the correct y (step 3) with probability at least $\frac{1}{2}$. Thus Vic will detect a cheating prover with probability at least $1 - 2^{-T}$.

Theorem 3.8: The above protocol is a zero-knowledge interactive proof system for the Discrete Log problem.

Proof: We will demonstrate this by describing the operation of a polynomial time machine M that will simulate the conversation between Peggy and any verifier V^*. For every round of the protocol, our simulator does the following:

M's algorithm.
Repeat at most $L = \frac{\log N}{\log 2}$ **times.**

1. Pick a random $c \in \{0\ 1\}$.
2. Pick a random $y \in \{0,...,N\text{-}2\}$.
3. Let $\gamma \equiv \alpha^y \beta^{-c} \bmod N$.
4. Compute b (the random bit generated by V^* in step 2 of the protocol) using V^*'s machine and save it. Let **b** be the intermediate results stored by V^* during the computation of b.
5. If $b = c$ then M outputs $\{\gamma, b, \mathbf{b}, y\}$ as required.

Until $b = c$.

If $b \neq c$ for all L iterations then M outputs "failure". Note that in the above $\alpha^y \equiv \gamma\beta^b \bmod N$.

Analysis

Case 1. N is prime. In this case, since ord<α>| |G| , where ord<α> is the order of the group generated by α, G is the finite group from which α is chosen and |G| = $\phi(N) = N-1$ which is the number of distinct elements from which y has been selected. Thus γ is uniformly distributed over <α> and γ and c are mutually independent. Also b is independent of c. Thus the probability that $b = c$ in at least one of the L executions steps 1-5 above is

$$\geq 1-\left[\frac{1}{2}\right]^{\frac{\log N}{\log 2}}$$

$$\text{Let } m = \left[\frac{1}{2}\right]^{\frac{\log N}{\log 2}}$$

$$\log m = \frac{\log N}{\log 2} \cdot \log 2$$

$$= -\log N = \log N^{-1}.$$

Thus the above probability is in fact $\geq 1 - \frac{1}{N}$ as required.

Recall that before the beginning of the protocol, both parties P and V are in an initial state and their work tapes contain certain initial data I_P^*. In addition, P's tapes also contains the secret x. Let $I_P = (I_P^*, x)$. Let $View_{P,V^*}(\alpha,\beta,N)$ denote the contents of V*'s work tapes after the protocol with P is over. This view represents a random variable whose probability distribution depends on the initial data I_P. Our simulator M produces a tuple $M(\alpha,\beta,N)$ with almost exactly the same probability distribution as $View_{P,V^*}(\alpha,\beta,N)$. Let Ω be the set of values which $M(\alpha,\beta,N)$ can take on, including the message "failure". Now, for every $\omega \in \Omega$ and $\omega \neq$ "failure", it is trivial that

$$\text{Prob}(\, M(\alpha,\beta,N) = \omega \mid M(\alpha,\beta,N) \neq \text{"failure"}) = \text{Prob}(\, View_{P,V^*}(\alpha,\beta,N) = \omega)$$

Also, as calculated above

$$\text{Prob}(M(\alpha,\beta,N) = \text{"failure"}) \leq \frac{1}{N} \text{ and hence}$$

$$S = \sum_{\omega \in \Omega} |\text{Prob}(M(\alpha,\beta,N) = \omega) - \text{Prob}(\text{View}_{P,V^*}(\alpha,\beta,N) = \omega)| \leq$$

$$\frac{1}{N} + \frac{1}{N} + 0 = \frac{2}{N}.$$

Case 2. N is not prime and $N=P_1P_2$ where P_1 and P_2 are primes of order $\sqrt{N}$. We note that the order of $<\alpha>$ in Z_N^*. ($\phi(N) = (P_1-1)(P_2-1)$) no longer divides $N-1$ (the number of elements from which y is selected). Hence, γ computed in step 3, is not uniformly distributed over $<\alpha>$, since y is chosen from $\{0,...N-2\}$ and $N-1$ is not a multiple of ord$<\alpha>$. However we get around this difficulty by restricting the above set from which we choose y to $\{0,...,\phi(N)-1\}$, consider conditional probabilities on this restriction and the argument used above still applies. Considering that

$$\text{Prob}(\phi(N) \leq y \leq N-2) = O(\frac{1}{\sqrt{N}})$$

it follows that S is bounded above by $O(\frac{1}{\sqrt{N}})$

The steps above are repeated T times thereby increasing the running time by that factor and the value of S by $\leq T$. But T is bounded above by a polynomial in Log N and hence our running time is still polynomial.❑

§3.9 Verifying Zero-Knowledge

In all the examples of zero-knowledge protocols presented so far, the prover attempts to convince the verifier that an input I belongs to some language L. In the following sections we extend this concept from *validation* that a given input $I \in L$, to *verification* that $I \in L$ or

$I \notin L$. In this new model, a prover P proves to a verifier V that $I \in L$ or $I \notin L$, such that V knows which claim is being established and is convinced with high probability. We also introduce the concept of *result-indisinguishability*, whereby a passive eavesdropper C monitoring the conversation between P and V cannot determine which claim is being proven. In fact, the protocol reveals *no knowledge at all* to an eavesdropping third party. The example protocol we present is due to Galil, Haber and Yung[GHY] and was the first non-trivial example of a language L for which proofs of membership and non-membership in L are done by means of the same protocol. More specifically, the predicate that will be established is that of being a quadratic residue or not. Recall that in previous examples, separate protocols were used for these two languages.

§3.9.1 Verifying Interactive Proofs and Zero-Knowledge

Suppose (P,V) is a pair of Interactive Turing machines and let $I \subseteq \{0, 1\}^*$ denote the set of valid inputs to the pair (P,V). Suppose that $L \subseteq I$ is a language for which P is able to compute the predicate $x \in L$. The definition for an interactive proof system used so far [GMR2] is now referred to as *validating* interactive proof system. We extend the definition to get the concept of a *verifying* interactive proof system for L and it is defined thus:

Given any Turing machine P^* interacting with V and common input $x \in I$, V halts with the correct value of the predicate $x \in L$ with high probability.

If (P,V) is a validating interactive proof system for a language L, then the definition used so far says that the system (P,V) is zero-knowledge if, given any probabilistic polynomial-time Turing machine V^*, there exists another probabilistic polynomial-time Turing machine M such that:

1. M can use V^* as a subroutine in the strong sense (see §2.3.3 Note 3).
2. $View_{P,V^*}(x,H)$ and $M(x,H)$ are polynomially indistinguishable

Note that an eavesdropper monitoring a successful execution of the protocol with input x learns with a high degree of probability that $x \in L$. We now extend the above definition to include a more general definition of zero-knowledge.

The operations of any pair of interactive Turing machines (P,V) define a partial function $f_{P,V}$ in the following way: Suppose P and V, upon common input x, use a total of at most k random bits, (with k is polynomial in $|x|$) during the course of their computations. For any k-bit string r, let $f_{P,V}(x,r)$ denote the result of the computation of P and V when the sequence of their coin flips is given by r. For the present discussion the function f will take on only Boolean values. More specifically, if (P,V) is a proof system for the language L, then $f_{P,V}(x,r)$ is (with a high degree of certainty,i.e for most r) equal to the membership bit ($x \in L$). We can now formulate a more general definition of zero-knowledge.

Definition (Verifying Zero-knowledge Proof) [GHY]: An interactive proof system (P,V) is said to be zero-knowledge if, given *any* probabilistic polynomial-time machine V^*, there exists another probabilistic polynomial-time Turing machine M such that:

1. Given any input x, M has one-time access to an oracle for the value of $f_{P,V}(x,r)$ for random $r \in \{0, 1\}^*$.
2. M can use V^* as a subroutine in the strong sense as described earlier.
3. $View_{P,V^*}(x,H)$ and $M(x,H)$ are indistinguishable (in polynomial time).

§3.9.2 Result Indistinguishability

An interactive proof system (P,V) is result-indistinguishable if an eavesdropper with access to the communication between P and V, upon common input x, gains no knowledge at all. More formally, the system (P,V) is result-indistinguishable if there exists a probabilistic polynomial-time Turing machine M such that $View_{P,V^*}(x,H)$ and $M(x,H)$ are indistinguishable (in polynomial time). It is worth noting that there is a significant difference between the machine M as described here and as described in our definition of zero-knowledge. While the latter has an access to an oracle for the value of $f_{P,V}$, the

former is not privy to such knowledge. In other words, M can simulate the conversation between P and V on input x regardless of the value of $f_{P,V}$ (even if such a value is the result of an intractable computation). Since this simulation is something that the eavesdropper could have computed for himself, no knowledge is gained even if such an eavesdropper is given a text of the conversation between P and V. Before presenting our protocol which is a verifying interactive proof system for the quadratic residuosity problem, we will first present a coin-flipping protocol due to Blum [Bl] which will allow two parties P and V to jointly generate a sequence of unbiased coin tosses.

§3.9.3 Blum's coin-flipping Protocol

Before giving the actual protocol, we will introduce some notation that will be used throughout the rest of this chapter. Let $\nu(N)$ denote the number of distinct prime factors of an integer N. Consider integers N of the form $N = \prod_{i=1}^{k} p_i^{e_i}$ such that for all i, $p_i^{e_i} \equiv 3 \bmod 4$

(where p_i are primes). The Jacobi symbol of some integer y mod N is defined as

$$\left(\frac{y}{N}\right) = \prod_{i=1}^{k} \left(\frac{y}{p_i}\right)^{e_i}$$

where $\left(\frac{y}{p_i}\right) = 1$ if y is a quadratic residue mod p_i and -1 otherwise (see Appendix B for further discussion).

Let **BL**, known as Blum Integers, denote the set of such integers. According to Blum [Bl] this set can be characterized in two alternate ways:

1. $N \in$ **BL** iff for any quadratic residue mod N half of its square roots have Jacobi symbol +1 and the other half have Jacobi symbol -1.

2. $N \in$ **BL** iff there exists a quadratic residue mod N which has 2 square roots with different Jacobi symbols.

We will consider a particular subset of **BL** defined as follows:

$\mathbf{BLS} = \{N : N \in \mathbf{BL}, N \equiv 1 \bmod 4 \text{ and } \nu(N) = 2\}$ or alternately

$\mathbf{BLS} = \{p^i q^j : p \neq q \text{ prime}, i,j \geq 1, p^i \equiv q^j \equiv 3 \bmod 4\}$

Finally let the language $\mathbf{I}$ be defined as

$\mathbf{I} = \{ (N,z) : N \in \mathbf{BLS}, z \in Z_N^*, (\frac{z}{N}) = +1\}$

and $\mathbf{L} = \{(N,z) : N \in \mathbf{I}: z \text{ is a quadratic residue mod } N\}$.

We now give Blum's coin flipping protocol.

Consider an integer N, such that $N \in \mathbf{BL}$ and $N \equiv 1 \bmod 4$. To generate a random bit b, P and V perform the following steps:

1. V picks a random $u \in Z_N^*$, computes $v \equiv u^2 \bmod N$ and sends v to P.
2. P picks $\sigma = \pm 1$ at random as a guess for $(\frac{u}{N})$, the Jacobi symbol of u and sends σ to V.
3. V sends u to P.
4. If $\sigma = (\frac{u}{N})$ then b:=1 else b:=0.

If factoring N is an intractable problem, then since $N \in \mathbf{BL}$, the protocol generates random bits by property 1 of Blum integers . We now demonstrate that the above protocol is zero-knowledge.

Consider a fixed, possibly dishonest Turing machine V^* interacting with P. We now describe a probabilistic polynomial-time Turing machine M whose output, upon input N, is a simulation of $\text{View}_{P,V^*}(N,H)$, namely the triple (v,σ,u) as defined in the protocol above. M also has access to an oracle which returns the value of the random bit b, and is allowed to use V^* as a subroutine.

M's algorithm

M consults the oracle and obtains the value of the bit b and executes the protocol with V^*. M lets V^* "send" v, simulates P's choice of σ in step 2 by flipping a coin and "receives" u from V^*. If the bit generated by this execution is b, then M outputs the triple (v, σ,u).

Otherwise, M resets V^* to its configuration at the beginning of the current round, goes to step 2, sends $-\sigma$ instead of σ to V^* and "receives" u'. M now outputs the triple $(v, -\sigma, u')$. In either case the triple output by M corresponds to the bit b and the distribution of its possible values is indistinguishable from $View_{P,V^*}(N,H)$. Note that if V^* is cheating then it may happen that u and u' are not the same. However, these must have the same Jacobi symbol, because the ability to extract 2 square roots of v (mod N) with different Jacobi symbols would enable V to factor N (for a proof see Appendix B).

§3.9.4 A Verifying Zero-Knowledge, Result-Indistinguishable protocol for the Quadratic Residuosity Problem

This protocol (P,V) will take inputs from the set I, as defined in the previous section, and give a verifying interactive proof system for the language L. Peggy will reveal to Vic the value of the predicate $x \in L$ without revealing any additional knowledge. Furthermore an eavesdropping third party will not be able to find out the value of the predicate. The protocol consists of 2 parts, the first being a validating interactive proof system for I, and the second (assuming that the first part is successfully completed) consisting of the actual protocol for the verifying proof system.

The first part of the protcol, establishing that the input string (N,z) belongs to I, is itself made up of three separate sub-protocols each of which validates a property of N or z.

1. $N \equiv 1 \bmod 4$, $\nu(N) > 1$, $z \in Z_N^*$ and $(\frac{z}{N}) = +1$
2. $N \in BL$
3. $\nu(N) \leq 2$

Stage 1: Trivial properties of N and z.

The verifier can easily check that $N \equiv 1 \bmod 4$, that N is not a prime power (this is done by first taking the q^{th} root x_q of N for all values of q up to $\log N$ using Newton's method. By

raising x_q to the power of q, we know that N is a prime power if we get back the original value), that $z \in Z_N^*$ (Euclid's Algorithm) and that $(\frac{z}{N}) = +1$. All of these can be accomplished in time polynomial in log N.

Stage 2: $N \in \mathbf{BL}$

This procedure due to Blum [Bl] depends on the first alternate characterization of Blum integers (see §3.9.3). To guarantee the correctness of the protocol with probability at least $1-\delta$, the protocol should be repeated k times where $k \geq \log\frac{1}{\delta}$. In order for this protocol to be executed, stage 1 must have been successfully executed so that $N \equiv 1 \bmod 4$.

Protocol

Repeat k times

1. Peggy, picks a random quadratic residue $r \in Z_N^*$ and sends r to Vic.
2. Vic picks random $\sigma \in \{1\ -1\}$ and sends σ to Peggy.
3. Peggy computes s such that $s^2 \equiv r \bmod N$ and $(\frac{s}{N}) = \sigma$, and sends s to Vic.
4. Vic checks to see if s meets the above conditions. If not, then he rejects the inputs and halts the protocol.

Stage 3: N has 2 prime factors

Let $Z_N^*(\pm 1)$ denote the set of elements of Z_N^* with Jacobi symbol ± 1 respectively. It can be shown that for N with exactly i prime factors, $\frac{1}{2^{i-1}}$ of the elements of $Z_N^*(+1)$ are quadratic residues. We will make use of this fact by having Peggy and Vic jointly pick random elements of $Z_N^*(+1)$ and Peggy will then go on to show that half of them are quadratic residues (by exhibiting their square roots mod N) thereby convincing Vic that $\nu(N) \leq 2$. Since it was established in stage 1 that $\nu(N) > 1$, together with $\nu(N) \leq 2$, this implies that $\nu(N)$ is *exactly* 2.

Peggy and Vic can jointly generate random elements of $Z_N^*(+1)$ by using Blum's coin-flipping protocol as described in the previous section. This requires that $N \in \mathbf{BL}$ and $N \equiv$

1 mod 4 which will be true if stage 2 has been successfully executed. To guarantee the correctness of the protocol with probability at least 1 -δ, it should be repeated k' times where $k' \geq \frac{16}{\delta}$.

1. Using Blum's coin-flipping protocol, Peggy and Vic generate k' random elements $r_1 \ldots r_{k'} \in Z_N^*(+1)$. This can be done by bitwise generation of elements of Z_N^* and discarding those with Jacobi symbol -1.

2. For each of the r_i's that are quadratic residues Peggy computes s_i such that
$$r_i \equiv s_i^2 \bmod N$$
and sends s_i to Vic.

3. Vic accepts the input if at least $\frac{3}{8}$ of the r_i's are quadratic residues and rejects otherwise.

Theorem 3.10: The above protocol is a zero-knowledge validating interactive proof system for the language L.

Proof: Each of the three stages is zero-knowledge and the concatenation of the three is also zero-knowledge. Since the main focus of this section is to describe the protocol by which the value of the predicate $x \in L$ is established in a result-indistinguishable manner, the reader is referred to [GHY] for a complete proof of the theorem.

As an aside, it is interesting to note that the third stage can also be used as a zero-knowledge proof system for the *value* of $\nu(N)$. This can be done by replacing the value $\frac{3}{8}$ by suitable constants. For instance if the prover can show $\frac{3}{8}$ of the r_i's to be residues the verifier should be convinced that $\nu(N) \leq 2$. If this fraction is between $\frac{3}{16}$ and $\frac{3}{8}$, the verifier should accept that $\nu(N) \leq 3$ and so on.

In conclusion we state that the concatenation of the 3 above stages yields a zero-knowledge proof system for L.□

§3.9.5 The Actual Protocol

At this point, assuming that the validating part of our protocol has been successful, we know with a high degree of certainty that the common input (N,z) satisfies the following properties:

1) $\nu(N) = 2$
2) $z \in Z_N^*$ and
3) $(\frac{z}{N}) = +1$

It is crucial for the next part of the protocol that these conditions hold. We can now describe the verifying, result-indistinguishable proof system for $\mathbb{L}$, where the inputs are taken from $\mathbb{I}$. In order to illustrate the different properties of the protocol, we present an initial version and subsequently refine it to include the desired properties.
Let $y \equiv -1 \bmod N$ and for any $x \in Z_N^*$ we define

$$\mathrm{Res}_N(x)=\begin{cases}1 \text{ if } x \text{ is a quadratic residue mod } N\\ 0 \text{ otherwise}\end{cases}$$

The following statements should be obvious. If we choose a random $r \in Z_N^*$ then

$x \equiv r^2 \bmod N$ is a random quadratic residue in Z_N^* (+1)

$x \equiv yr^2 \bmod N$ is a random quadratic non-residue in Z_N^* (+1). (This is because if $N \in BL$ and $N \equiv 1 \bmod 4$, then $y \equiv -1$ is always a quadratic non residue. If another quadratic residue is desired, Peggy can use a subprotocol to prove to Vic that y is indeed a quadratic non residue.)

$x \equiv zr^2 \bmod N$ is one of the above depending on whether z is a quadratic residue mod N or not.

In the following protocol the parameter k should be chosen such that $k = \Omega(\log\frac{1}{\delta})$ to guarantee the correctness of the protocol with probability at least $1-\delta$.

Version I

Repeat $3k$ times

1. Vic picks a random $r \in Z_N^*$.

2. Vic picks a random $c \in \{1\ 2\ 3\}$ and

   Case c of:

   1: $x \equiv r^2 \bmod N$

   2: $x \equiv yr^2 \bmod N$

   3: $x \equiv zr^2 \bmod N$

   and sends x to Peggy.

3. Peggy computes the value

   $b = \mathrm{Res}_N(x)$

   and sends b to Vic.

4. Vic checks if the following conditions hold:

   i) If $c = 1$ then $b = 1$

   ii) If $c = 2$ then $b = 0$

   iii) if $c = 3$ then the value of b should be consistent with any previous values of b for which $c = 3$

   If any of the above conditions is violated Vic halts and rejects.

If the conditions stipulated in step 4 are satisfied for all iterations, Vic accepts the value of $\mathrm{Res}_N(x)$ to be the value of b he receives from Peggy when he sends $c = 3$.

In the above protocol, there are 2 ways in which a dishonest prover P^* could fool our verifier Vic.

1. P^* could try to convince V that z is not a quadratic residue when it in fact is.

2. P^* could try to convince V that z is a quadratic residue when it in fact is not.

The only way for P^* to accomplish case 1 would be to correctly guess among all the iterations for which Vic has sent a quadratic residue whether the latter is a c=1 or c=3

residue. Similarly, to achieve case 2, P^* would have to distinguish between non residues for which $c = 2$ and those for which $c = 3$. The probability of either event occuring is 2^{-kC} for some suitable constant C. Hence the above protocol is a *verifying* interactive proof system for L. However, the result-indistinguishability property does *not* hold. This is because an eavesdropper on the conversation can determine the value of $Res_N(x)$ by keeping a tally of the bits b sent during a conversation. The value of the most frequently occuring bits gives the value of $Res_N(x)$.

Version II

The following simple modification to the above protocol, however, enables us to achieve result-indistinguishability: Before Peggy starts the protocol she flips a fair coin to decide between the following

$R(x) = Res_N(x)$ or,

$R(x) = 1 - Res_N(x)$

as an encoding for the value of the bit b sent to Vic during step 3 of the protocol. During step 4 Vic now checks for consistency in the following way. Vic should receive the same bit b during all case-1 iterations, its complement in case-2 iterations and a consistent value for case-3 iterations which indicates whether z is a quadratic residue or not. If during any iteration Vic realizes that the value of b contradicts the above pattern, he halts the protocol and rejects. Thus the above remains a verifying interactive proof system for L. Furthermore, we have also achieved result-indistinguishability since an eavesdropper monitoring the conversation observes a value of b two-thirds of the time and its complement the rest of the time but its value gives no information whatsoever as to the value of $Res_N(z)$.

The version presented so far is not zero-knowledge. This is because Peggy wants to avoid acting as a residuosity oracle for Vic. This is essentially the same problem encountered in the previous protocol presented for quadratic non residues. Vic could send specific values for x in step 2 of the protocol instead of constructing them as specified and in this manner

would extract information from Peggy which Vic could not have computed for himself. The only difference between the above protocol and the ones previously presented is that we know have 3 possible classes instead of just 2. In [GHY], a somewhat cumbersome scheme was devised to ensure that Vic does follow the protocol and constructs the x's as specified. However, this can be considerably simplified using cryptographic capsules [Co] as was done in the Quadratic Non residue protocol of §3.3.

The key addition to the protocol of §3.3 is the addition of a third set of possibilities. The prover now chooses from three sets X (c=1), Y (c=2), and Z (c=3). Members of X consist of randomly generated residues (class 0), those of Y are randomly generated non residues (class 1), and members of Z are the product of random residues and z and will thus be of class 0 or 1 depending on whether or not z is a residue mod N.

By using 3-component capsules, the protocol of [GHY] becomes considerably simplified. Vic simply prepares a master capsule C consisting of one member each from the above three sets and say 100 additional capsules of the same form. Peggy now designates a random subset of these capsules which Vic opens, thereby demonstrating that they were constructed as specified. Vic now shows that each remaining capsule is of the same form as C by matching components and showing that their quotients are residues. Peggy should now be fully convinced that C was generated as required and can now tell Vic which of the capsules components is different from the others thereby revealing to Vic the class of z. The rest of the protocol remains unchanged.

Theorem 3.11: Given input belonging to I, the above protocol is a result-indistinguishable zero-knowledge verifying interactive proof system for L.

Proof: Since we have already shown that version II is a verifying proof system for L we simply have to show that the refinement preserves that property. Since components of the capsules are randomly generated it is impossible for a cheating P^* to distinguish between c=1 and c=3 iterations when z is a quadratic residue and c=2 and c=3 iterations when z is a

quadratic non residue and hence the protocol remains a verifying interactive proof system for L.

We prove the zero-knowledge property by describing the computation of a probabilistic Turing machine M which will simulate the conversation between our prover Peggy and an arbitrary, possibly cheating, verifier V^*. M consults the oracle and learns the value of $Res_N(z)$. M then flips a fair coin to simulate Peggy's choice of whether to encrypt as $R(x)= Res_N(x)$ or, $R(x) = 1 - Res_N(x)$. The remainder of the simulation is very similar to the protocol for quadratic non residues as presented in §3.3 and is left as an exercise for the interested reader. We must now prove that the protocol is result-indistinguishable by describing the operation of a probabilistic Turing machine M' which will simulate the conversation between Peggy and an arbitrary verifier V^*. However, in this case M' does not have access to the oracle and does *not* know the value of $Res_N(z)$. M' flips a coin to decide whether to simulate the choice $R(z) = 0$ or $R(z) = 1$. M' now simulates the computations of Peggy and V^* with the following exceptions

- In step 2, M' chooses $x \equiv zr^2$ with probability $\frac{2}{3}$ and $x \equiv zyr^2$ with probability $\frac{1}{3}$.
- In step 4, M' outputs $b = R(z)$ if $x \equiv zr^2$ and $b = 1 - R(z)$ if $x \equiv zyr^2 \bmod N$.

In this way, the values of x output by M' have the same distribution as those output by V^* in a real conversation between Peggy and V^* and $M'(N,z)$ is polynomially indistinguishable from $View_{P,V^*}(N,z)$.❑

Chapter 4
Zero-knowledge Protocols for all languages in NP

§4.1 Introduction

In the previous chapter we presented a number of examples of zero-knowledge interactive protocols for a variety of languages. However, each of the languages described had a different protocol. In the following chapter we present methods by which general protocols can be built for various languages in NP and beyond.

§4.2 All languages in NP have zero-knowledge proof systems: The Graph 3-colouring method

Theorem 4.1: If f(.;.) (as defined in §3.6) is a secure probabilistic encryption scheme, then every language in NP possesses a zero-knowledge interactive proof system

Proof[GMW2]: We proceed by construction in 4 steps. Let L be any language in NP. We construct a zero-knowledge proof system by incorporating a fixed reduction to graph 3-colourability. Both parties compute the 3-colourability instance from the common input and Peggy demonstrates to Vic via the zero-knowledge protocol of §3.4 that this instance is 3-colourable. However, in the above theorem, Peggy has infinite computing power and hence can always compute the appropriate colouring. The next theorem considers the more realistic case where both Peggy and Vic are limited to polynomial-time computations.❑

Theorem 4.2: If secure probabilistic encryption schemes exist, every language in NP has a zero-knowledge interactive proof system where the prover is a probabilistic polynomial-time machine which gets an NP proof as auxiliary input.

Proof[GMW2]: The problem with this scenario is that Peggy might not be powerful enough to play her role in the protocol. If she were given a colouring of the 3-colourability instance then following the protocol of §3.4 would be easy. However, she is only given a NP proof for membership in an arbitrary language L in NP. We can get around this

difficulty by using the following Karp [GJ] reductions to transform L to a graph 3-colourability instance.

1. Transform L to a SAT instance S via Cook's Theorem [C].
2. Transform S to a 3SAT instance 3S via Cook's Theorem.
3. Transform 3S to a graph 3-colourability instance G via [GJ].
4. Prove via the protocol of §3.4 that G is 3-colourable.

The important thing to note about the above reductions from L to G is that they all preserve the witnesses (these are the values which satisfy the original equation) from the original instance to the reduced instance so that the prover is always able to fulfill her role in the protocol.❑

§4.3 A Zero-knowledge Protocol for Boolean Satisfiability

A protocol for Boolean satisfiability was first proposed in [BC1] which relied substantially on particular properties of quadratic residues and hence did not extend to arbitrary encryption functions. Chaum [Ch] proposed a similar protocol but under a different model where the prover is restricted to polynomial-time computations but which allowed the verifier to have infinite power. This model also stressed the unconditional privacy of the prover's secret. Finally Brassard and Crépeau [BC2] proposed a model in which all parties involved have "reasonable" computing power. The difference amongst these various models can be illustrated by considering the case where Peggy claims to know the factorization of some integer n.

- In the [GMR2] model it is pointless for her to try to convince Vic of this since he already knows that this is so due to Peggy's infinite computing power. However, the [GMR2] model is interesting for its theoretical implications.
- In the setting of [Ch] Peggy's secret factors cannot possibly be unconditionally secure once she makes n public. She might as well just convince Vic that she knows the factors by giving them to him as Vic, with his infinite computing power, will always be able to

factor n. However, Peggy might want to demonstrate that she knows something without revealing what she knows. For example, if her claim had been that she knows non-trivial divisors of n where n is the product of several primes, it would make sense to use the setting of [Ch] to convince Vic of her knowledge without disclosing any information about which of the divisors she knows even if Vic has infinite computing power.

• In the setting of [BC2] where all parties have reasonable computing power it is reasonable for Peggy to attempt to convince Vic of her knowledge without revealing any information that would enable Vic to compute the factors of n. Of course, in this setting, if Peggy and Vic have similar computing power, the obvious question is how did Peggy obtain a hard enough proof to be of interest to Vic? One answer is that Peggy was lucky or worked hard enough to find it. However, in our case Peggy could simply have picked some random large primes and multiplied them together to produce n. She then knows the factors of the result even though she is no better than Vic at factoring large integers. Abadi *et al.* [AABFH] give a theorem on the efficient generation of solved hard instances in NP.

We next present the results of Brassard, Chaum and Crépeau [BCC] in which all the above concepts are unified. They consider the resources available to either party during and after the protocol. Their main result is a protocol that is unconditionally secure for both parties as long as Peggy is unable to factor some large integer (or find a discrete logarithm) *while the protocol is taking place.* Once the protocol is over, it is too late for either party to try to cheat regardless of their computing power. We contrast this with the protocol of §3.4 which depends on the inability of the verifier to invert some encryption function. Moreover, this weakness is retroactive in that even if Vic is unable to do this while the protocol is taking place, he could go back to old transcripts of Peggy's proofs and spend as much time as he likes in deciphering them.

§4.3.1 Bit Commitment.

The crux of the protocols to be presented is the notion of *bit commitment.* This is the process by which Peggy is able to commit to the value of some bits in a way that prevents Vic from learning about them without Peggy's help. Intuitively, if Peggy wants to commit to a bit *b* she places *b* in a box, locks it and sends the box to Vic. Vic can only learn the value of the bit if Peggy gives him the key. The main primitive used in [BCC] for implementing bit commitment is the *blob.* Each blob is used by Peggy to commit to either a 0 or a 1. The following are the abstract defining properties of blobs.

1 (Completeness). Peggy can commit to blobs representing 1 and blobs representing 0.

2 (Soundness). Peggy can open any blob she has committed to and can convince Vic of the value of the bit she ,in effect, committed to when she committed to the blob. Thus, there is no blob that Peggy can open both as 0 and as 1.

3. (Security). When presented with a blob, Vic cannot tell which bit it represents. This remains true even after other blobs have been opened.

4. Blobs do not carry side-information in the sense that the processes by which Peggy commits to and opens blobs are not related to any secrets she wishes to keep from Vic.

It will be useful to think of the above properties in the following way. Peggy commits to a bit (property 1) by writing it on the floor and before Vic can see it she covers it with a piece of opaque tape. Although Vic cannot tell which bit is under the tape (property 3), Peggy can no longer change it. Peggy "opens" the blob by letting Vic remove the tape and look at the bit (property 2). A subtle point worth noting is that it is not necessarily the case that a given blob must encode a unique bit. It is not the blob itself that determines the bit but rather Peggy's knowledge about it. In this case the closer analogy is the box example that was used above. In the following protocol we assume the existence of blobs and later give

implementations of blobs that yield different levels of security for the prover and the verifier.

§4.3.2 The Basic Protocol

Assume that Peggy knows the satisfying assignment to some Boolean formula. The basic protocol allows Peggy to convince Vic that she knows such an assignment without revealing any information about it. Consider the following formula

$$\psi = [\,(p \wedge q) \oplus (\overline{q} \vee r) \wedge \overline{[\,(\overline{r} \oplus q) \vee (p \wedge \overline{r})]}$$

and let <p=true, q=false, r=true> be Peggy's secret satisfying assignment. Note that although finding the satisfying assignment to the above formula is trivial, as the number of variables in the formula increases the cost of testing all possible combinations increases exponentially.

§4.3.2.1 The Initial Set Up

Peggy and Vic initially agree on the layout of the Boolean circuit to compute ψ. Figure 4.1 gives the circuit for ψ as well as Peggy's satisfying assignment and truth table of each gate. The outlined rows correspond to the circuit's computation on Peggy's satisfying assignment. Seeing the outlined rows is sufficient to verify that ψ is satisfiable by simply checking the consistency of each wire and making sure that the output of the final wire is a 1. The idea of the protocol is for Peggy to prove that she knows how to outline one row in each truth table without revealing any information about which rows they are.

§4.3.2.2 Overview of the Protocol

The protocol will consist of *k* rounds in which the following is done:

1. Peggy scrambles the circuit's truth tables and commits to a corresponding set of blobs.

2. Vic then issues one of the following two possible challenges

A) Vic asks Peggy to show that the blobs really encode a valid scrambling of the circuit's truth tables.

B) Vic asks that Peggy open the blobs that correspond to rows that would be outlined assuming that the scrambling is valid.

The challenges are thus designed in such a way that Peggy could only meet both of them if she knows the satisfying assignment but answering any one of them yields no information about it. Thus if Peggy does not know the satisfying assignment and attempts to cheat she will be caught in any round with probability $\frac{1}{2}$. By repeating the above steps k times the probability that Vic is fooled by a cheating Peggy is at most 2^{-k}, which is equivalent to an exponential increase in security for a linear increase in the number of rounds.

§4.3.2.3 The Scrambling Process

The scrambling of each truth table will consist of random row permutations and column complementations. Figure 4.2a shows the truth table for the Boolean conjunction "AND" of Figure 4.1. The rows of the truth table are randomly permuted to yield the table in Figure 4.2b. Any of the 24 possible permutations may be chosen with equal probability. One bit is then randomly chosen for each column of the truth table and each column is complemented if the bit for that column is a 1 as illustrated in Figure 4.2c. Note that we can still recognise the scrambled truth table as being the Boolean conjunction provided the complementation bits shown in the circles are specified. It is important to be consistent and that all truth table columns corresponding to the same wire in the circuit be either all complemented or remain the same. Figure 4.3 gives the result of random permutations and complementations. After producing a circuit similar to that of Figure 4.3 Peggy commits to it as follows: for each truth table bit, Peggy commits to a blob that she knows how to open. She also keeps the complementation bits secret.

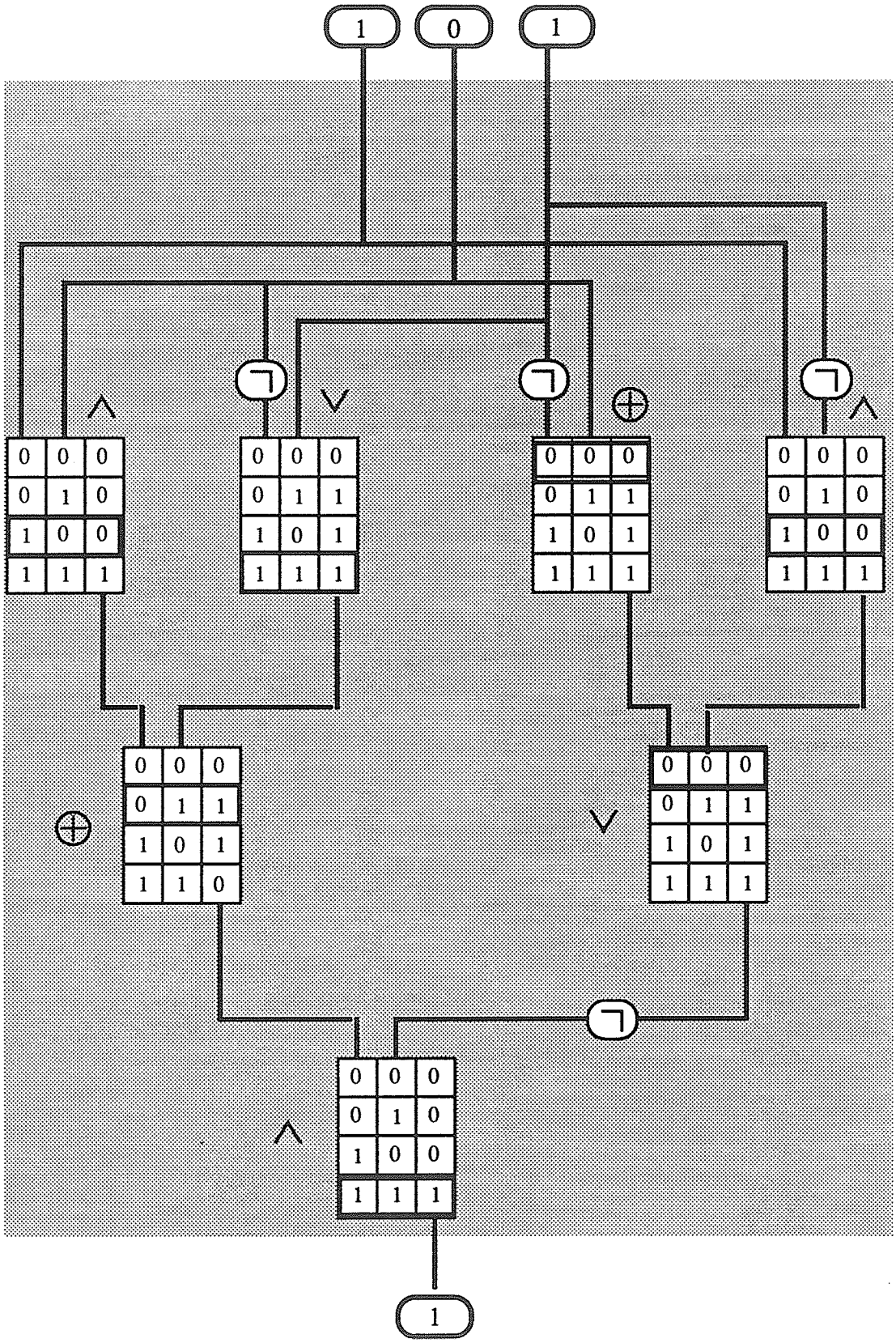

Fig 4.1. A Boolean Circuit with explicit truth tables and rows outlined

Going back to the floor and tape analogy, Peggy has now written Figure 4.3 on the floor and covered each bit with opaque tape before Vic can see it.

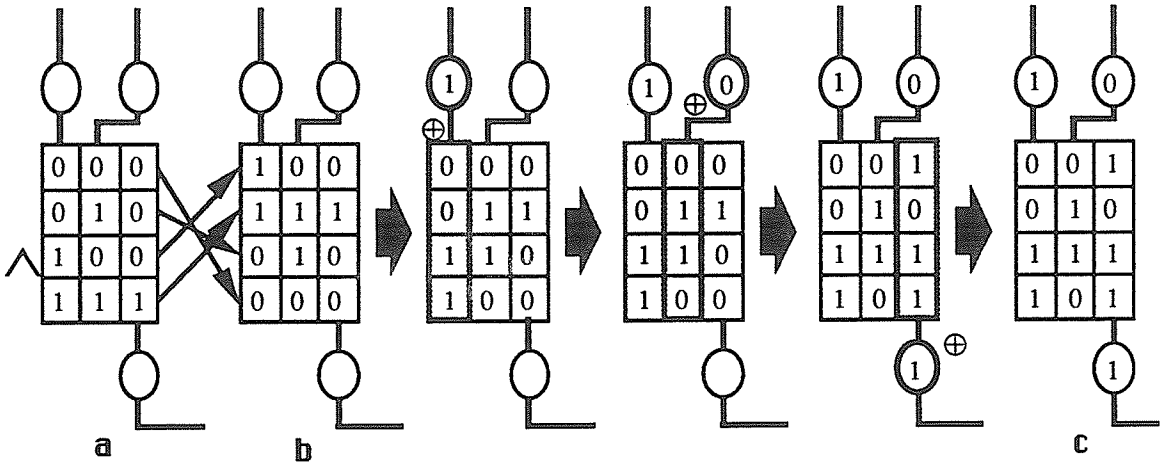

Fig 4.2. Permutation and complementation of a truth table.

§4.3.2.4 The Challenge

Peggy having committed herself, Vic may at this point issue one of either Challenge A or B as described above.

- If Vic's challenge is A, Peggy must open each and every blob she just committed to as well as reveal the complementation bits used in the scrambling process. Intuitively, Peggy strips off all the tape on the floor and allows Vic to look at the equivalent of Figure 4.3. This allows Vic to verify that the information concealed by the blobs indeed encodes valid permutations and complementations of the circuit's truth tables.
- If Vic's challenge is B, Peggy only opens those blobs corresponding to one row in each truth table. The rows that will be opened are those that were outlined in Figure 4.1 in their now (probably) new location as determined by the row permutations.

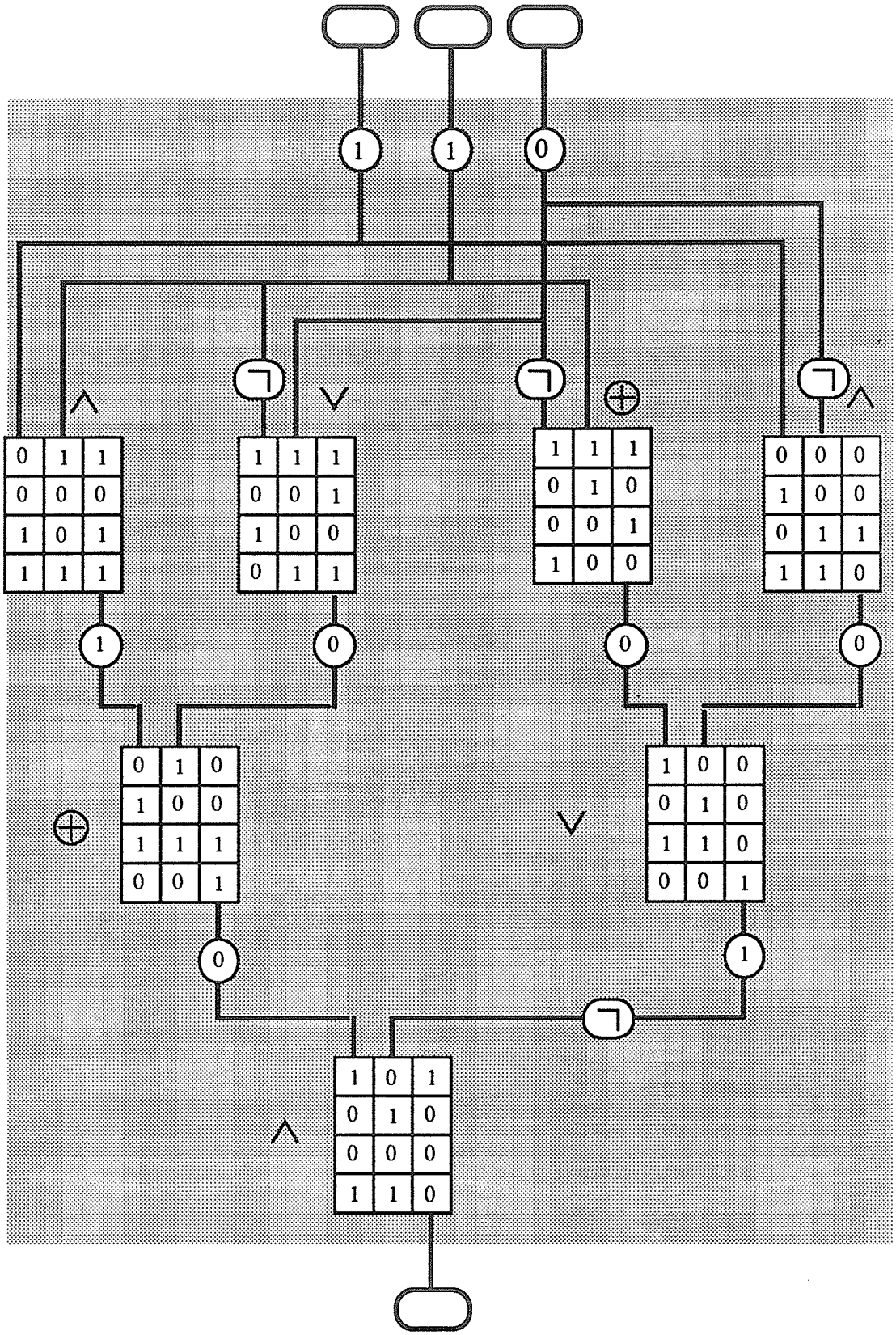

Fig 4.3. A circuit with randomly permuted and complemented truth tables.

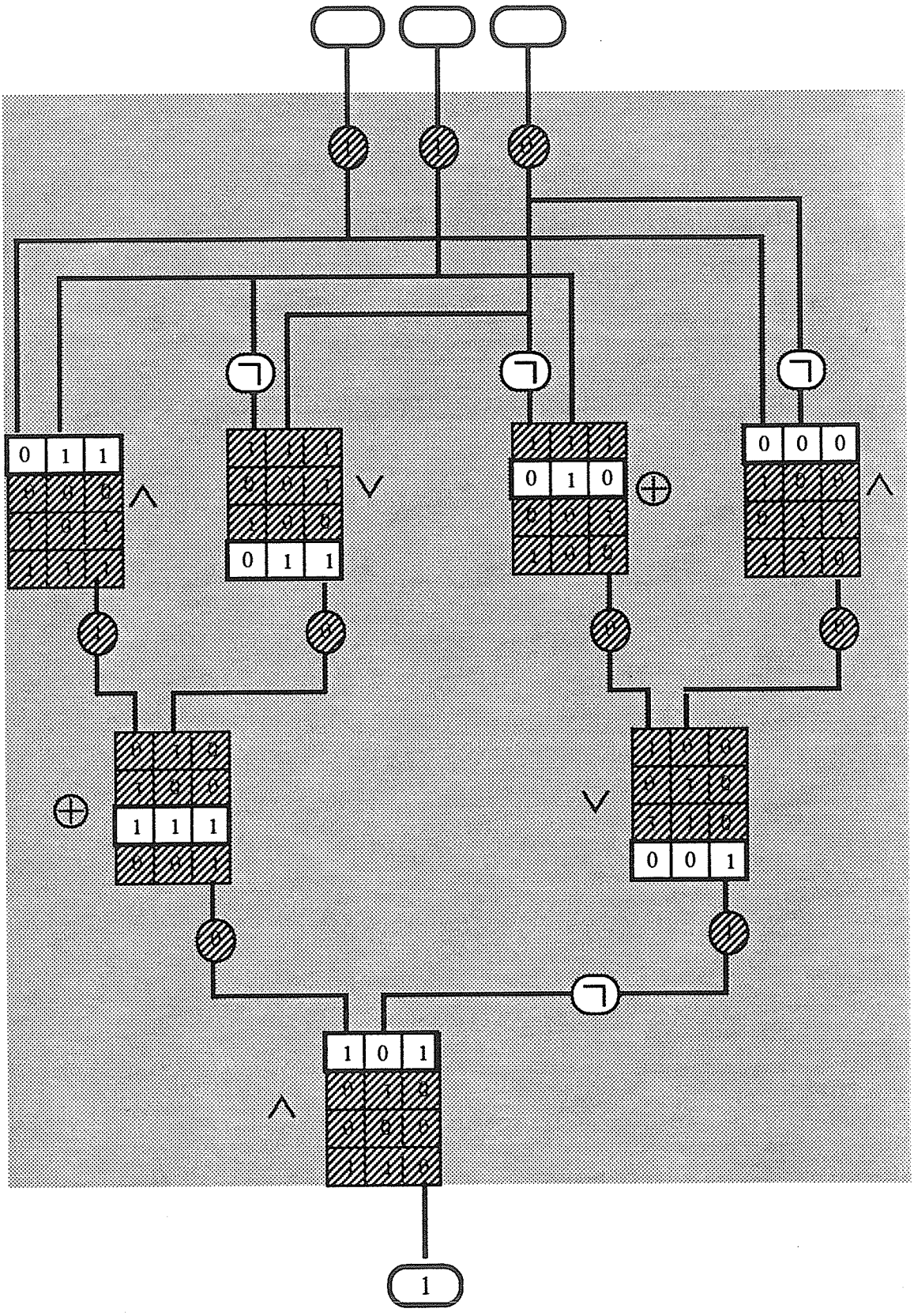

Fig 4.4. The existence of a satisfying assignment is revealed.

Intuitively, Peggy strips off pieces of tape to reveal to Vic the equivalent of Figure 4.4. This allows Vic to verify the consistency of each wire and the fact that the final output is indeed a 1.

§4.3.3 Proof of Correctness of the Protocol

In order for the protocol to be correct the following 3 requirements should be satisfied except perhaps with an exponentially small probability.

1. Peggy can carry out her share of the protocol if she knows a satisfying assignment to ψ.
2. If Peggy does not know a satisfying assignment for ψ, no matter how she pretends to follow the protocol, Vic will detect the cheating.
3. If Peggy knows a satisfying assignment for ψ, and she follows her share of the protocol, nothing is revealed to Vic that will enable him to determine the satisfying assignment (or even partial information about it) even if he deviates arbitrarily from the protocol.

We now show why and how the above requirements are met.

Requirement 1. Peggy is able to commit to blobs and open them as required owing to defining properties 1 and 2 of blobs. Anyone can randomly permute rows and complement columns of a truth table to obtain a Figure 4.3. Since Peggy knows the satisfying assignment she can outline the appropriate rows in each of the scrambled truth table by simply remembering which columns are complemented and where each permutation has taken each row that she knows would have been outlined in the original truth table. Thus requirement 1 is met.

Requirement 2. Assume Peggy does not know the satisfying assignment. In any round she can either commit to genuine permutations and complementations of the original circuit's truth tables or she can commit to something phoney. In the former case she cannot answer challenge B without knowing a satisfying assignment for ψ and in the latter case

she cannot answer challenge A without breaking defining bit property 2 of blobs. Thus, as long as Peggy is unable to predict Vic's challenges (and hence construct an arrangement that would enable her to answer the challenge) she will be caught cheating by Vic with probability $\frac{1}{2}$ in each round. In a k-round protocol Peggy will be caught cheating with probability at least $1 - 2^{-k}$ and thus requirement 2 is satisfied.

Requirement 3. This is argued in two steps as there are two levels of information that may be released in the protocol.

1. We first argue that Vic cannot learn anything about the satisfying assignment (beyond the fact that Peggy knows it) simply from receiving a Figure 4.3 or Figure 4.4. If Vic issues challenge A, he receives a Figure 4.3 which consists of randomly permuted and complemented versions of the originally agreed circuit's truth tables. This obviously does not help Vic to find Peggy's satisfying assignment as he could have produced such a figure by himself even if ψ was not satisfiable. On the other hand, if Vic issues challenge B, he receives Figure 4.4 which is equivalent to applying a true one-time pad (Peggy's independent wire complementations) on the Boolean values carried by the circuit's wires during the computation of a satisfying assignment. A property of one-time pads is they hide all information so that no advantage is gained by Vic that would enable him to learn something about the satisfying assignment. The only way for Vic to obtain additional information about the satisfying assignment would be to obtain a matching Figure 4.3 (which contains the complementation bits) and Figure 4.4 (which reveals the position that the random permutation placed the satisfying row), something that Peggy will obviously never release.

At this point, although we have established the correctness of the protocol, the latter may not in fact be zero-knowledge. It is quite possible that only Peggy has the required technology to commit to blobs in which case the verifier obtains something that he could not have produced on his own. The protocol is only zero-knowledge if the the third

requirement is strengthened to the effect that Vic cannot learn *anything at all* beyond the fact that Peggy knows the satisfying assignment. Recall that a protocol should only be considered zero-knowledge if and only if Vic can reproduce on his own the conversations that he would have had with Peggy during a real execution of the protocol, simply after being told by a trusted oracle that the formula is satisfiable. This is possible if we strengthen defining property 4 of blobs to read as follows:

4' (Simulatability). Vic can simulate what he would have been provided in the process by which Peggy commits to blobs that she could open as 0 and those she could open as 1. He is further able to simulate the process by which Peggy would have opened these blobs had she committed to them herself.

The blobs are said to be *simulatable* if in addition to properties 1, 2, and 3, they also satisfy property 4' above.

Theorem 4.3: If simulatable blobs are used, then our protocol is zero-knowledge.

Proof: We describe the process by which Vic could simulate the conversations that he would have had with Peggy in a real execution of the protocol. Note that since Vic does not know the satisfying assignment, he will fail each round with probability $\frac{1}{2}$ during the course of the simulation. For the cases resulting in failure, Vic simply discards them and repeats the current round with new random choices. We next describe the simulation process.

§4.3.4 The Simulation

1. Vic flips a fair coin to determine which of challenges A or B he will be prepared to answer.
2. Depending on the outcome of the coin toss, he generates a Figure 4.3 or a Figure 4.4 as required.

3. Vic prepares a collection of blobs corresponding to whichever of Fig. 4.3 or 4.4 he has produced by simulating the process that Peggy would have used. He can do this because of the simulatability of blobs as defined above.

4. At this point, Vic asks himself (honestly!) which challenge he would have issued had this been a real conversation with Peggy.

5. If the challenge corresponds to one that he can actually meet he simulates Peggy opening the blobs.

   else

   Vic fails, in which case he "rewinds" himself to the point immediately before the current round and starts again with new random choices. Thus the probability that any round will have to be repeated t times is 2^{-t}.

Due to defining property 3 of blobs, there is no correlation between the challenge Vic decides he is ready to meet and the one that he actually issues to himself. After k successful rounds have been executed, the random variable representing the simulated view is indistinguishable from that representing Vic's view during a conversation with the real prover. Thus our protocol is a zero-knowledge protocol for SAT.❑

§4.3.5 The Parallel Version

Some interesting points arise when we consider the parallel version of the basic protocol in which all k rounds of the protocol are carried out simultaneously. In this scenario, Peggy commits herself at the outset to blobs corresponding to k circuits similar to Figure 4.3. Vic then sends to Peggy his string of challenges and Peggy opens the blobs in the manner requested by the challenges. Such a version might be more desirable for efficiency reasons.

This, however, makes it possible for Vic to choose his challenges as a function of the entire collection of blobs. Although this does not allow him to obtain any information about Peggy's satisfying assignment, it might allow Vic to subsequently convince others that ψ is

satisfiable by showing them the transcript of his conversation with Peggy. This even holds if simulatable blobs are used. Assume that blobs are bit strings and Peggy commits to a blob by showing it in the clear. Vic can cheat in the following manner. After receiving blobs corresponding to the k circuits with randomly permuted and complemented truth tables from Peggy, Vic concatenates them together and uses the result as an input to some one-way function. (A one-way function, f, is an irreversible function such that the computation of $c = f(m)$, given m, is easy; but for a given c, it is computationally infeasible to determine m). He then uses the first k bits of the output to determine the k challenges to be issued. For this version to be zero-knowledge, Vic must be able to simulate the conversation that he would have had during a real execution of the protocol. However, if Vic attempts to use the simulation technique outlined above to simulate the protocol, the probability that the one-way function will actually yield as the first k bits of its output exactly those bits that correspond to the challenges he is able to meet, is exponentially small so that the parallel version is not zero-knowledge. Thus, although the parallel version is *not* zero-knowledge and does not reveal any information about Peggy's secret, its transcript may be used to convince others of the existence of Peggy's secret! If it is important that the protocol be carried out in parallel, we can make the protocol zero-knowledge by further strengthening property 4 of blobs so that in addition to properties 1-3 they also satisfy the following:

4* (trapdoor or chameleon). Vic can simulate what he would have been provided in the process by which Peggy commits to blobs. Furthermore, for each of these blobs, Vic can simulate either the process by which Peggy would open it as a 0 or the process by which Peggy would open it as a 1.

Such blobs are known as *chameleon* (or *trapdoor*) and allow Vic to do exactly what defining property 2 of blobs had prevented Peggy from doing. A trapdoor blob is such that there is a secret known as the *key to the trapdoor* that allows Vic to open the blob to reveal a 0 or 1 as required. They also allow Vic to simulate his conversations with Peggy without

ever encountering any failures even if he deviates arbitrarily from the actual protocol. This allows Vic to simulate the parallel version because since the blobs are trapdoor, Vic does not need to have already decided in which way he expects Peggy to be able to open them, since they can be opened to reveal either a Figure 4.3 or a Figure 4.4. In order to simulate the parallel version, Vic simulates Peggy's commitment to as many blobs as she would use. He then chooses his challenges as if the blobs actually came from Peggy. Whenever he chooses Challenge A, he randomly permutes and complements the Boolean circuit's truth tables to produce something like Figure 4.3 and opens the blobs accordingly. Whenever he chooses Challenge B, he randomly selects a random row in each truth table and one Boolean value for each wire in the circuit (except for the final output wire for which he chooses 1) and opens the blobs in these rows to produce something like Figure 4.4.

§4.4 Zero-knowledge Protocols for all Languages in NP: The Circuit Satisfiability Method.

In the previous sections we have shown how to prove the existence of zero-knowledge protocols for all languages in NP by reducing them to the graph 3-colourability method. In this section we will proceed by reducing all languages in NP to a circuit satisfiability problem

Theorem 4.4: Assuming the existence of bit commitment schemes, all languages in NP possess an interactive zero-knowledge protocol. (Note that these protocols are not technically interactive proof systems in the sense of [GMR2] which allowed the prover to be infinitely powerful and hence allows her to cheat by changing her commitments. As we will see in the next chapter these commitments typically rely on cryptographic assumptions. Brassard and Crépeau [BC4] have argued that when the verifier's faith in the prover's claim relies precisely on such assumptions, the protocols should instead be referred to as *arguments*. Interested readers are referred to [BC4] for further elaborations)

Proof: Using the fact that satisfiability is NP-complete [C,GJ], the basic protocol can be used to produce zero-knowledge protocols for all languages in NP. Let $L \subseteq \Sigma^*$, where Σ represents $\{0,1\}$. From the definition of NP, there exists a "proof system" $Q \subseteq L \times \Sigma^*$ such that whenever $x \in L$, there exists a succinct "certificate" c to that effect, and one can efficiently verify that c is valid proof that $x \in L$. We say that such a c is verifiable information to the effect that $x \in L$. Using Cook's theorem [C], Peggy and Vic can both construct from any $x \in L$, a Boolean formula $\psi_L(x)$ that is satisfiable if and only if $x \in L$. Due to the constructive nature of Cook's theorem it is enough for Peggy to know some succinct c such that $<x, c> \in Q$ in order to efficiently deduce a satisfying assignment for $\psi_L(x)$. Therefore for any $x \in L$, and $L \in NP$ where Peggy knows a succinct certificate c to the effect that $x \in L$, she can use the basic protocol to convince Vic that $\psi_L(x)$ is satisfiable and hence that $x \in L$ and she knows how to prove it.❑

As noted by several researchers and formalised by Feige, Fiat and Shamir [FFS] the term "zero-knowledge" is somewhat misleading since the prover does reveal one bit of information, namely the value of the predicate $x \in L$ and hence some researchers have suggested the term *minimum* be used. It is possible to extend the concept to "knowledge about knowledge" where the prover reveals that she knows the *status* of x with respect to L. Tompa and Woll [TW] also considered this concept using a somewhat different approach than [FFS]. Consider the restricted set of languages $L \in NP \cap$ co-NP. In this case we can construct for each $x \in \Sigma^*$ two Boolean formulae $A_L(x)$ and $B_L(x)$ such that exactly one of them is satisfiable ($A_L(x)$ if $x \in L$ and $B_L(x)$ if $x \notin L$). The disjunction $C_L(x) = A_L(x) \vee B_L(x)$ is always satisfiable. Assume Peggy knows the value of the predicate $x \in L$, and she knows the corresponding succinct NP certificate. This gives a satisfying assignment for either $A_L(x)$ or $B_L(x)$, whichever is satisfiable and hence she can also satisfy $C_L(x)$. Using the basic protocol she is able to convince Vic that she knows the satisfying assignment for $C_L(x)$. This does not reveal anything about x to Vic but it does convince him that Peggy knows the status of x with respect to L and that she can prove it.

§4.5 Practical Zero-knowledge:Going Beyond NP

In the previous sections we have presented general zero-knowledge protocols for languages in NP where both interacting parties are restricted to polynomial-time computations. However, the class of languages that can be recognised by such protocols extends beyond NP. In the next section we will characterize the class of languages for which such protocols exist.

We first define some important classes of languages, give some important results with respect to these classes and show how to extend our protocol to languages outside of NP.

Babai [Ba] introduced the notion of an Arthur-Merlin protocol. In this model, an infinitely powerful Merlin plays the role of the prover, Arthur that of the verifier and Merlin sees all of Arthur's coin tosses. This implies that any communication between the two parties will consist of only random strings. It is pointless for Arthur to send anything else since Merlin can obviously compute for himself whatever Arthur is able to send.

Definition: A language L is said to be in AM[k] if there exists an Arthur-Merlin k-round protocol, that is k alternating messages between Arthur and Merlin, Arthur sending first such that

1. If $x \in$ L, Arthur accepts with probability $> \frac{2}{3}$
2. If $x \notin$ L, Arthur accepts with probability $< \frac{1}{3}$

We let MA[k] be defined as above except that Merlin sends first. Babai [Ba] showed that MA[k] $\subseteq$ AM[k]. Recall that a language L is said to be in IP if it can be recognised by interactive proof system (see §2.2).

Although this new definition appears more restrictive, Goldwasser and Sipser [GS] show that the AM and IP are equivalent with respect to language recognition. They even show that for any language having an interactive protocol in R rounds, we can find an AM protocol requiring R + 2 rounds. In fact it is shown in [GS] that a language has an interactive proof system if and only if it has an AM proof system. Furthermore Babai [Ba]

showed that for every constant k, AM[k] collapses to AM[2]. Fortnow [F] shows that if a language L $\in$ IP and possesses a perfectly zero-knowledge proof system, L's complement has a constant round interactive proof system. Ben-or *et al.* [BGGHKRM] have shown that every language in IP has a zero-knowledge proof system. The above results are all theoretically compelling but most of them not only allow but *require* that the prover be infinitely powerful. For example, in the graph nonisomorphism protocol (see §3.3.2) the prover must be able to decide graph isomorphism and Fortnow's result depends on this model in a critical way. Thus it is important to consider the class of "practical IP", introduced by Brassard and Damgaard [BD], where "practical IP" refers to the class of languages that can be recognised when both Peggy and Vic are restricted to polynomial-time computations.

Definition [G]: A language L is said to be in BPP if there is a probabilistic polynomial time algorithm which on each input can determine membership in the language with a small probability of error.

It is reasonable to consider BPP as the real class of tractable problems since the error can always be decreased below any threshold $\delta > 0$ by repeating the algorithm $\alpha \log \delta^{-1}$ times and taking the majority answer, where the constant α only depends on the original probability of error. We can now define the class of languages (called "Practical IP" in [BD]) which can be recognised by zero-knowledge protocols when both parties are restricted to polynomial-time computations.

Babai's class MA is equal to the set of combined languages from NP and BPP (i.e. MA = NP$\cup$BPP) and Brassard and Damgaard [BD] prove that "practical IP $\subseteq$ MA". The class MA is defined exactly as NP, except that we are satisfied with a BPP algorithm for deciding, given x and c whether $\langle x,c \rangle \in$ Q. Whenever $\langle x,c \rangle \in$ Q we now refer to c as a *convincing argument* for the fact that $x \in$ L. It cannot be called a certificate because in general it cannot be verified with certainty. If $x \in$ L where L $\in$ MA it is enough for Peggy to know a convincing argument to that effect. Consider the set B of integers having exactly

two prime factors. If Peggy generates two distinct integers p and q that pass a probabilistic primality test to her satisfaction she is convinced that $n = pq \in B$ with $\langle p,q \rangle$ as her convincing argument. We next show how to extend the basic protocol to consider languages $L \in MA$.

§4.5.1 The MA Protocol [BCC]

We consider languages $L \in MA$. Assume Peggy possesses a succinct convincing argument c to the effect that $x \in L$. Since c is not a certificate Peggy is not certain that $x \in L$. Our protocol allows us to convince Vic that $x \in L$ and she knows a convincing argument such that Vic does not obtain any information about it.

§4.5.1.1 Preliminary Step

Peggy and Vic agree on the error probability δ that they are willing to tolerate for the certifying BPP algorithm. They consequently modify the algorithm so that its error probability does not exceed δ. Once this is done we can now assume that the probability of error of the certifying algorithm is negligible.

Let $n = |x|$ and $m = |c|$ where the value of m is uniquely determined as a known function of n. This is so that the protocol will not have to hide the value of m from Vic.

Let r be an upper bound on the number of coin flips that the certifying algorithm can perform on any input $\langle x,\hat{c} \rangle$ where $\hat{c}$ is of size m. Using an argument similar to the proof of Cook's theorem we can get a Boolean formula ψ with at least $m+r$ variables. If the first m variables are set to represent the binary string c, and the next r variables are set randomly, then (except with probability at most δ) it is easy to set the remaining variables (if any) that will satisfy ψ if and only if c is a convincing argument that $x \in L$. Knowing x and the certifying algorithm makes construction of the formula possible so that it can be made public. However, the basic protocol cannot be used in its present form as Vic cannot trust Peggy to actually choose the r inputs truly at random. By the same token, Peggy cannot allow Vic to choose them either as a judicious choice of these values might enable

Vic to learn something about Peggy's convincing argument c. This is resolved by requiring that Peggy and Vic perform a sub-protocol that will generate random values that cannot be influenced by either party. The sub-protocol should also be performed in such a way as to prevent Vic from learning the value of the random bits. Although this could be done by using Blum's [Bl] protocol for coin-tossing "in a well", the same goal can be achieved if we require that blobs satisfy the following additional properties:

Property 5 (Equality). Given 2 unopened blobs that encrypt the same bit that Peggy has committed to, Peggy is able to convince Vic that she could open them to reveal the same bit without revealing any additional information.

Property 6 (Inequality). Given 2 unopened blobs that encrypt different bits that Peggy has committed to, Peggy is able to convince Vic that she could open them to reveal different bits without revealing any additional information.

We can now use property 6 to implement coin-flipping as follows:

Coin-Tossing in a Well

1. Peggy commits to 2 blobs that she could open to reveal 2 distinct bits.
2. She convinces Vic that this is so and asks him to pick a blob.
3. When Vic makes his choice, the coin toss is determined and its outcome is the bit Peggy could show by opening the blob chosen by Vic.

However, Vic cannot tell the value of the bit unless Peggy opens the corresponding blob which, of course, she will never do. We can now describe the general protocol for the case of probabilistically verifiable information. The idea is similar to the protocol already presented with a few modifications to take care of the further complexities discussed above. We now describe the modifications.

§4.5.1.2 The Extended Protocol

Peggy and Vic have agreed on the Boolean circuit that will be used to probabilistically verify Peggy's convincing argument c that $x \in$ L. Peggy commits to m blobs, each

corresponding to one bit of c, and the two parties perform the coin-tossing protocol described above to generate the r random bits corresponding to the r remaining variables of the Boolean formula represented by the circuit. The basic protocol must be changed to force Peggy to use the proper bits for the inputs corresponding to c and to the r random bits all the while ensuring that no information is leaked to Vic that would enable him to learn something about the value of these bits. We use our earlier example to illustrate how this would be done. Assume Peggy has committed to some blob b that she could open to reveal a 1 but Vic does not know this. Peggy now wants to convince Vic that she knows a satisfying assignment to ψ for which the first variable corresponds to the blob she could open as b. Peggy scrambles the circuit's truth tables as before to produce something similar to Figure 4.3 and commits to it. For each input bit that she has committed to (the first bit in our example; the first $m+r$ bits in general) Peggy now also commits to the complementation bits that were used to produce the current Figure 4.3.

If Vic issues challenge A, Peggy reveals the equivalent of Figure 4.3 exactly as before by opening all the corresponding blobs as well as the blobs corresponding to the complementation bits. Vic can then check that the blobs encoded a valid scrambling of the truth tables. However, if Vic issues challenge B, Peggy must do more than reveal the equivalent of Figure 4.4. This is because Figure 4.4 says nothing about the value of the input variables chosen by Peggy. Vic wants also to be convinced that these were chosen properly by Peggy. For example the wire corresponding to the first input is 0 (see first bit in outlined row of the top left truth table of Figure 4.4) and Peggy uses the equality property of blobs to prove to Vic that blob b and the blob associated with the corresponding wire complementation encode the same bit without revealing what the bit is, of course. If that value had been a 1, Peggy would have used the inequality property to convince Vic that the blob b and the blob associated with the corresponding wire complementation encode different bits. This process is repeated for all $m+r$ variables and everything else remains the same as the basic protocol.

The above protocol is however not zero-knowledge from a theoretical standpoint. This is because different convincing arguments may cause the certifying algorithm to fail with different probabilities. This element of randomness prevents Vic from simulating exactly the conversation that would take place in a real execution of the protocol with Peggy. Although running the protocol an exponentially large number of times with Peggy could enable a very powerful Vic to learn something about Peggy's secret, the protocol is still very safe if δ is chosen to be small enough.❑

§4.6 Summary

Throughout this chapter we have described general methods for zero-knowledge protocols for all languages in NP and BPP. However these can be generalised further. Convincing a certain verifier that a circuit is satisfied by an input with a given blob encryption is a special case of the following problem: If a circuit computes a certain function $F(I) = O$, then convince a verifier that this is true given a blob encryption of I and O. Boyar and Peralta [BP] give improved zero-knowledge protocols for arithmetic operations and give general methods for proving satisfiability of circuits containing both arithmetic and logical gates.

Chapter 5
Bit Commitment and General Results

§5.1 Introduction

In the previous chapter we showed how zero-knowledge protocols can be built for various languages in NP and beyond using two different methods. However, the existence of bit commitment schemes was simply assumed. It is clear that bit commitment is central to the protocols presented and the necessity for secure schemes is thus critical. In this chapter we will describe various bit commitment schemes and then discuss the security obtained from different implementations. We also discuss some general results about bit commitment and finally present some efficiency improvements for the practical implementation of zero-knowledge protocols.

§5.2 Blob Implementations

Blobs can be implemented in many ways. When they are based on cryptography and computational complexity they are necessarily imperfect. These imperfections manifest themselves in many ways. If it is impossible for the prover to change her commitments after having committed herself, the implementation is said to be *unconditionally secure for the verifier*. Conversely if it is impossible for the verifier to determine the value hidden by a blob the implementation is said to be *unconditionally secure for the prover*. If impossibility is replaced by "near impossibility" then the scheme is said to be *statistically secure* for the prover or the verifier whichever the case might be. By nearly impossible, we mean that the probability of the event occuring is exponentially small and even infinitely powerful parties cannot influence the outcome. Thus, the event occurs purely by luck. We can now give a more formal definition, which we borrow from [BCC], of bit commitment. A *bit commitment scheme* consists of 2 sets X and Y along with an efficiently computable verification function $v: X \times Y \rightarrow \{0, 1, \bullet\}$ where $\bullet$ stands for undefined. In order to

commit to some bit $b \in \{0, 1\}$, Peggy picks a pair $x \in X$ and $y \in Y$ such that $v(x,y) = b$. Thus x is the blob and y is Peggy's additional knowledge about it. Peggy commits to b by giving x to Vic and opens the blob by giving y to Vic who can then compute $v(x,y)$ and learn the value of b.

§5.2.1 Blobs Statistically Secure for the Prover

In the following section we describe a blob implementation that is statistically secure for the prover.

§5.2.1.1 Based on the Presumed Difficulty of Factoring [BC2]

Recall from §3.2.1 that QR_n denotes the set of all quadratic residues mod n. Let $y \in Z_n^*$, if $x \equiv y^2 s$ it is impossible to distinguish an x produced when $s = 1$ from that produced when $s \in QR_n$. Let $n = pq$ where p and q are 2 distinct large primes. Given n and $s \in QR_n$ it is infeasible at the present time to compute a square root of s mod n unless the factors of n are known.

Blob Generation Protocol

1. Vic randomly chooses two distinct large primes p and q and forms the product n=pq.
2. He then picks a random $t \in Z_n^*$, $(t^2 > N)$ and computes $s \equiv t^2$ mod n and sends n and s to Peggy.
3. Vic assumes temporarily the role of the prover and uses another zero-knowledge protocol to prove to Peggy that $s \in QR_n$ and that he knows one of the square roots.

Once the steps 1-3 above are successfully executed the blobs can then be defined as two sets X and Y where $X = QR_n$ and $Y = Z_n^*$ and

$$v(x,y) = \begin{cases} 0 \text{ if } x \equiv y^2 \bmod n \\ 1 \text{ if } x \equiv y^2 s \bmod n \\ \circ \text{ otherwise} \end{cases}$$

Peggy commits to some bit b by picking a random $y \in Z_n^*$, computing $x \equiv y^2 s^b \bmod n$ and giving x to Vic. She keeps y as a secret witness which allows her to open the blob. Thus, the completeness property of blobs holds.

Since any quadratic residue could be used to encrypt zeros as well as ones, Vic cannot tell an encryption of a 0 from an encryption of a 1 and hence the security property holds. Soundness holds computationally since the only way Peggy could open a blob to show either a 0 or a 1 would be to obtain a square root of s. This would allow Peggy to simply send $y^2 s$ as her x and send y if she wants to open the blob as 1 and send $y\sqrt{s}$ if she wants to open the blob as a 0. Since Peggy is restricted to polynomial time computations we assume this to be infeasible for her.

Claim: The above blobs are statistically secure for Peggy.

Proof: Although Vic cannot tell the bits hidden by a blob, there exists a subtle way that he could cheat. In order to this he would have to use an s that is a quadratic nonresidue but he would have to be able to convince Peggy that it is in fact a residue during step 3 above. If he succeeds and the probability of such an event is exponentially small, then blobs that encrypt 0 will always be quadratic residues mod n, something that Vic can easily check as he knows the factors of n. This would permit Vic to obtain additional information about Peggy's secret. Thus these blobs are statistically secure for Peggy, as only luck can help Vic obtain additional information since no amount of computing power (since a quadratic residue is used to represent both a 0 and a 1) will help him in doing so. Peggy could ask Vic to reveal a square root of s at the end of the protocol which would convince her that Vic had not learned any of her secrets. It is clear that such blobs possess the trapdoor property with t being the key that allows Vic to open blobs as both 0 and 1.❑

Claim: The above blob implementation makes retroactive (off-line) cheating impossible.

Proof: Consider the case where an efficient factoring algorithm is discovered some time after the protocol between Vic and Peggy takes place. Since quadratic residues are used to encrypt both 0 and 1, Vic will still have no idea of which case Peggy had in mind when she produced the blobs.❑

Claim: If Peggy does not actually know the secret, she can only fool Vic by obtaining a square root of *s while the protocol is taking place.*

Proof: If Peggy manages to obtain such a square root of *s* at the outset, then she can open any blob as 0 or 1. However she must be able to do this before the end of the first round in which she is asked a challenge that she is not prepared to answer. Obtaining a square root of *s* at a later time is of no use to her.❑

§5.2.2 Blobs Unconditionally Secure for the Prover

§5.2.2.1 Based on the Discrete Log Problem [CDG, BKK]

For a discussion of the discrete log problem see §3.5.1. We can use the intractability assumption of the discrete log problem to create blobs provided we strengthen the problem to the effect that computing the discrete log modulo a large prime *p* remains infeasible even if the factors of *p-1* are known.

Blob Generation Protocol

1. Peggy and Vic agree on a prime number *p* for which they both know the factors of *p-1*.
2. They also agree on the generator α of the multiplicative group Z_p^*.
3. Using their knowledge of the factors of *p-1* they can both verify with certainty that *p* is a prime and α is a generator.

4. Vic chooses a random $s \in Z_p^*$ $(s \neq 1)$ and gives it to Peggy. Due to the intractability assumption of the discrete log problem Peggy is unable to compute e such that $s \equiv \alpha^e \pmod{p}$.

Once the steps 1-4 above are successfully executed the blobs can then be defined as two sets X and Y where $X = Z_p^*$ and $Y = \{0,1,2,....p\text{-}2\}$ and

$$v(x,y) = \begin{cases} 0 \text{ if } x \equiv \alpha^y \bmod p \\ 1 \text{ if } x \equiv \alpha^y s \bmod p \\ \bullet \text{ otherwise} \end{cases}$$

Peggy commits to some bit b by picking a random $y \in Y$, computes $x \equiv s^b \alpha^y \bmod p$ and gives x to Vic while keeping y as a secret witness which allows her to open the blob and hence property 1 of blobs holds. Since any element of X could be used to encrypt zeros as well as ones, Vic cannot tell an encryption of a 0 from an encryption of a 1 and hence property 3 holds. Property 2 holds computationally since the only way Peggy could open a blob to show either a 0 or a 1 would be to obtain the value of e (which we assumed is infeasible for her). Such blobs possess the trapdoor property with e being the key that allows Vic to open blobs to reveal 0 or 1 as required.

Claim: The above blobs are unconditionally secure for Peggy.

Proof: The above blobs are fundamentally different from those in the previous section. This is because there is not even a possibility for Vic to cheat. This follows from the fact that blobs that Peggy could open as 0 and those she could open as 1 are indistinguishable depends only on the fact that p is prime and that α generates Z_p^*. Both of these facts can be verified by Peggy even *before* the protocol starts.❑

Claim: The above blobs make retroactive cheating impossible.

Claim: If Peggy does not know the secret she can only cheat by finding the discrete log of s while the protocol is taking place.

Proof: The proof of the above two statements is analogous to those in the previous section and is not repeated here.

§5.2.3 Blobs Statistically Secure for the Verifier

§5.2.3.1 Based on the Quadratic Residuosity Problem [BC1]

Recall from §3.6.4 that $Z_n^*[+1]$ denotes those elements of Z_n^* with a Jacobi symbol 1. If $n = pq$ where p and q are distinct large primes, half the elements of $Z_n^*[+1]$ are quadratic residues and half are non residues modulo n.

Blob Generation Protocol

1. Peggy chooses two distinct large primes and forms the product n.
2. She also chooses a quadratic nonresidue s with Jacobi symbol +1.
3. She convinces Vic that n has only 2 prime factors using the protocol outlined in stage 3 of §3.6.4 or the method outlined in [PG].
4. She convinces Vic that s is indeed a quadratic nonresidue mod n using the protocol of §3.2.2 or §3.6.4.

Using the quadratic residuosity assumption we assume that Vic cannot distinguish random quadratic residues from nonresidues mod n with Jacobi symbol +1.

Once the steps 1-4 above are successfully executed the blobs can then be defined as two sets X and Y where $X = Z_n^*[+1]$ and $Y = Z_n^*$ and

$$v(x,y) = \begin{cases} 0 \text{ if } x \equiv y^2 \bmod n \\ 1 \text{ if } x \equiv y^2 s \bmod n \\ \circ \text{ otherwise} \end{cases}$$

Peggy commits to some bit b by picking a random $y \in Y$, computes $x \equiv y^2 s^b \bmod n$ and gives x to Vic while keeping y as a secret witness which allows her to open the blob and hence property 1 of blobs holds. Property 2 holds unconditionally if and only if x is a

quadratic residue. Thus there is no blob that Peggy could open as both a 0 and a 1. Property 3 holds computationally as we assume that testing for quadratic residuosity is infeasible for Vic.

Claim: The above blob implementation in statistically secure for Vic.

Proof: The only way for Peggy to cheat, if she does not know the secret, is to use an s that is a quadratic residue and attempt to convince Vic that it is in fact a quadratic nonresidue. This can only happen with an exponentially small probability. If she were successful in doing it she could open any blob either as 0 or 1. Since luck is the only way she could achieve this and no amount of computing power will help her, we consider these blobs to be statistically secure for Vic.❑

Claim: The above blob implementation makes retroactive cheating possible.

Proof: An algorithm capable of factoring efficiently would enable Vic to obtain additional information about Peggy's secret as there is no ambiguity as to which blobs encrypt which bit value.❑

§5.2.4 Blobs Unconditionally Secure for the Verifier

§5.2.4.1 Based on the Discrete Logarithm Problem [BCC]

Let p be a large prime, α be the generator of Z_p^* and u be the smallest integer such that 2^u does not divide $p-1$. Peralta [Pe] has shown that given any $s \in Z_p^*$, it is easy to compute the u-1 least significant bits of the unique e such that $0 \leq e \leq p-2$ and $s \equiv \alpha^e \bmod p$. Under the intractability assumption of the discrete log problem it is infeasible to learn anything about the u^{th} significant bit of e which is as difficult as finding the discrete log itself [Pe].

Blob Generation Protocol

1. Peggy and Vic agree on a p and α exactly as in §4.6.1.2 and let u be as above.

   Once the above step is successfully executed the blobs can then be defined as two sets X and Y where $X = Z_p^*$ and $Y = \{0,1,2,....p-2\}$ and

$$v(x,y) = \begin{cases} y_u \text{ if } x \equiv \alpha^y \bmod p \\ \\ \circ \text{ otherwise} \end{cases}$$

where y_u is the u^{th} significant bit of y.

Peggy commits to some bit b by picking a random $y \in Y$ such that $y_u = b$, computes $x \equiv \alpha^y \bmod p$ and gives x to Vic while keeping y as a secret witness which allows her to open the blob and hence property 1 of blobs holds. Note that in this case Peggy must remember y as she could not recompute it from x. Property 2 of blobs holds unconditionally as α is a generator of Z_p^* and hence the discrete logarithm of x is uniquely defined so that there is no blob that Peggy could open both as 0 or 1. Property 3 holds computationally if the discrete log assumption is strengthened to the effect that finding discrete logarithms mod p is still infeasible even if the factorization of $p-1$ is known.

Claim: The above blob implementation in unconditionally secure for Vic.

Proof: As opposed to the previous implementation it is now no longer possible to cheat by an argument similar to that of §5.2.2.1

Claim: The above blob implementation makes retroactive cheating possible.

Proof: An algorithm for solving the discrete log problem would enable Vic to obtain the hidden bits encrypted by the blobs and hence enable him to obtain additional information about Peggy's secret.

§5.2.5 Trapdoor Blobs not Based on Cryptographic Assumptions

The bit commitment schemes presented in the previous sections relied on specific cryptographic assumptions, namely the difficulty of factoring and the discrete log problem. In the following sections we present two bit commitment schemes that have the trapdoor property but do not rely on such assumptions. These blobs are instead based on computational complexity.

§5.2.5.1 Based on Graph Isomorphism [BC2]

G is defined to be a hard graph if, given G and a random isomorphic copy of G, it is computationally infeasible to find an isomorphism between the two graphs.

Blob Generation Protocol

1. Peggy and Vic agree on some hard graph $G_0(V,E_0)$.
2. Vic randomly chooses a random permutation π and produces a graph $G_1(V,E_1)$ where $(u,v) \in E_1$ iff $(\pi(u),\pi(v)) \in E_0$.
3. Vic gives H to Peggy and convinces her using the zero-knowledge protocol of §3.3.1 that G_0 and G_1 are isomorphic to each other. The roles are again temporarily reversed as Vic assumes the role of the prover.

Once the steps 1-3 above are successfully executed the blobs can then be defined as two sets X and Y where $X = \{K(N,E^*) \mid K$ is a graph isomorphic to $G_0\}$ and $Y = \{\gamma \mid \gamma: V \rightarrow V$ where γ is a permutation) and

$$v((V,E^*),\gamma) = \begin{cases} 0 \text{ if } (u,v) \in E^* \text{ iff } (\gamma(u),\gamma(v)) \in E_0 \\ 1 \text{ if } (u,v) \in E^* \text{ iff } (\gamma(u),\gamma(v)) \in E_1 \\ \bullet \text{ otherwise} \end{cases}$$

That is, Peggy commits to a bit b by picking a random permutation $\gamma \in Y$ and producing K, a random isomorphic copy of G_b and keeps γ as a witness that allows her to open the blob. Since K is isomorphic to both G_0 and G_1 and Vic knows the isomorphism he is unable to figure out which case Peggy had in mind when she produced the blob so that any K can be used to encrypt either a 0 or 1 and hence the security property holds. The soundness property holds computationally as the only way Peggy could open a blob as 0 or 1 is for her to obtain some isomorphism between G_0 and G_1 which we assumed was infeasible for her. Such blobs possess the trapdoor property with π being the key that allows Vic to open blobs to reveal 0 or 1 as required.

Claim: The above blobs are statistically secure for Peggy.

Proof: In the above implementation Vic could cheat by using some graph G_1 that is *not* isomorphic to G_0 and manage to convince Peggy in step 3 that it in fact is. If he succeeds, and the probability of success is exponentially small, Peggy's blobs K will always be isomorphic to G_0 when they encrypt 0 and to G_1 when they encrypt 1, a fact that can be verified by Vic. Thus, these blobs are statistically secure as only luck can help Vic and no amount of computing power can help him figure out Peggy's secret.

§5.2.5.2 Based on the Directed Hamiltonian Graph Problem [FeS1]

This scheme is based on the zero-knowledge protocol for DHC given in §3.6 and arises from the following observation. In step 3 of the protocol, if Peggy does not know a cycle in G, she cannot answer both a $b = 0$ and a $b = 1$ challenge.

Blob Generation Protocol

1. Vic selects a Hamiltonian graph G with n vertices, sends it to Peggy and proves to her via the protocol of §3.6 that he knows a cycle in G.
2. Peggy commits to a zero by choosing a random permutation π, permutes the vertices of G and commits (using a non-trapdoor bit commitment scheme) to the entries of the adjacency matrix of the resulting graph and commits to a one by choosing the n vertex clique and committing to its adjacency matrix (which is all ones).
3. Peggy opens a blob to reveal a zero by sending π and opens the blobs that encrypted the entries of the matrix so that Vic can verify that they indeed correspond to a valid permutation of G. She opens a blob to reveal a 1 by opening a random cycle in the adjacency matrix.

The scheme is trapdoor as knowledge of a cycle in G allows Vic to open as 1 blobs that he had originally committed to a 0. The above protocol is secure for Vic as long as Peggy is unable to find a cycle in G while the protocol is taking place and is secure for Peggy if Vic

cannot break the non trapdoor commitment scheme. Such schemes can be constructed from any one-to-one one-way function [GL] and from one-way functions at the expense of an extra move [ILL]. A non trapdoor commitment scheme based on any pseudo-random generator is given by Naor [N]. The most general result so far is due to Impagliazzo and Luby [IL] who have argued that the existence of one-way functions is a prerequisite for any protocol whose security relies on computational complexity.

§5.2.6 Oblivious Transfer and ANDOS

An oblivious transfer (OT) protocol allows Vic to obtain Peggy's bit b with probability $\frac{1}{2}$. At the end of the protocol Vic knows whether or not he received the bit and Peggy does not. ANDOS ("All-or-Nothing Disclosure of Secrets") is a tool invented by Brassard, Crépeau, and Robert [BCR1]. In this scenario, Peggy owns n secret strings $s_1,s_2....s_n$. An ANDOS protocol allows Vic to choose any s_k from Peggy in such a way that prevents her from learning the value of k. That is, Peggy gives away a string s_k but doesn't know which one. Furthermore, as soon as Vic learns the value of s_k he has wasted his chance of learning anything about the remaining strings.

§5.2.6.1 Blobs based on Oblivious Transfer [Cr]

To commit to a bit b, Peggy picks a vector $\mathbf{b} = b_1,b_2...b_n$ such that $b = b_1 \oplus b_2 \oplus ... \oplus b_n$ and sends the b_i's to Vic using an oblivious transfer protocol. At the end of the protocol Vic will have received on the average half of the bits in $\mathbf{b}$ and he knows which ones they are whereas Peggy does not. At this point nothing is revealed about the value of b unless Vic obtains all the bits of $\mathbf{b}$ which will happen only with exponentially small probability. In order to open the blob Peggy sends Vic the vector $\mathbf{b}$. Vic checks that all the bits that he received correctly actually correspond to the respective bits in $\mathbf{b}$. Other blob implementations based on oblivious transfer are given in [BCC, K].

§5.2.6.2 Blobs based on ANDOS

Peggy asks Vic to prepare two secret strings S_0 and S_1. To commit to a bit b, Peggy obtains S_b through an ANDOS protocol. As mentioned above Vic then has no idea which string Peggy chose and hence nothing is revealed about the value of b. To reveal the value of b Peggy simple shows the string S_b to Vic. Note that with OT blobs Peggy is the sender and Vic the receiver whereas the roles are reversed for ANDOS blobs. The relationship between OT and ANDOS has been investigated in [Cr, BCR2] who showed that any OT protocol can be efficiently transformed to an ANDOS protocol. Thus in the ANDOS protocol, if the underlying OT protocol is unconditionally secure for Peggy (Vic) then the ANDOS protocol is unconditionally secure for Peggy (Vic).

§5.2.7 Quantum Blobs [BC3]

These blobs are based on the principles of quantum cryptography [BB]. The basic idea is to use polarized photons to transmit bits and relies on Heisenberg's Uncertainty Principle to prevent cheating and hence infinite computing power is of no help (even if P = NP) in breaking the scheme. An experimental prototype using this principle has been built and achieved the secure transmission of a secret key over a public channel [BBBSS]. Such a system is unconditionally secure for both parties. It is in fact shown that Vic can cheat only if he can build a device that can transmit information faster than the speed of light.

§5.2.8 Notes on Bit Commitment Schemes

In this section we present some general results and observations on bit commitment schemes. As we have seen, the existence of bit commitment schemes implies the existence of zero-knowledge protocols and are thus central to such protocols.

If the bit commitment scheme used in a zero-knowledge protocol is unconditionally secure for the prover, the protocol will be perfectly zero-knowledge in that a simulation will

produce the same probability distribution as the actual protocol. If the scheme is only statistically secure for the prover then the protocol is only statistically (almost perfectly) zero-knowledge. Impagliazzo and Luby [IL] have shown that bit commitment schemes unconditionally secure for the verifier exist if and only if one-way functions exist. Since such schemes are only computationally secure for the prover, protocols resulting from their use are only computationally zero-knowledge. On the other hand, Brassard and Yung [BY] have shown that if one-way group actions (a generalization of one-way group homomorphisms introduced by Impagliazzo and Yung[IY] and used by Brassard, Crépeau, and Yung [BCY] for implementing bit commitment schemes) exist, then bit commitment schemes unconditionally secure for the prover can be implemented which yield perfectly zero-knowledge protocols.

Some bit commitment schemes also have additional properties, a particularly important one being the trapdoor property which allows the parallelization of zero-knowledge protocols. Consequently, the use of trapdoor blobs allows the construction of *constant* round zero-knowledge protocols for all languages in NP [BCY, FeS1].

§5.3 Constant round Perfect Zero-knowledge Protocols [BCY, FeS1]

Brassard, Crépeau, and Yung [BCY] have constructed a 6-move perfect zero-knowledge protocol for all languages in NP using a bit commitment scheme based on the discrete logarithm problem. Under the same assumption, Feige and Shamir [FeS1] have constructed a 4-move protocol which is perfectly zero-knowledge. Under the more general assumption that one-to-one one-way functions exist they build a protocol that still has 4 moves but is now only computationally zero-knowledge. Under the still more general assumption that one-way functions exist, the protocol becomes a 5-move computationally zero-knowledge protocol. (This is because the construction of schemes based on one-way functions requires a preliminary move [N,ILL].) We next outline the process by which [FeS1] built their constant round zero-knowledge protocol.

Zero-knowledge protocols are usually constructed by sequentially iterating a basic step n independent times with each round increasing the confidence of the verifier. The obvious way to obtain a constant round protocol would be to execute the n steps in parallel thereby minimizing interaction. Although a parallel execution doesn't reveal anything about Peggy's secret, it does allow Vic to convince others of the existence of her secret. But we have seen that using trapdoor blobs allows the construction of parallel zero-knowledge protocols (§4.3.5). Thus using trapdoor blobs would seem to solve the problem but this introduces a further complication. How does Peggy know that the commitment scheme used by Vic is indeed trapdoor? For example, if Vic uses the bit commitment scheme based on the DHC (§5.2.5.2) how can Peggy be sure that the graph G that Vic sends contains a cycle *and* that Vic knows it. Thus, Vic must prove to Peggy his knowledge of the trapdoor information. This can be done by a temporary role reversal during which Vic proves to Peggy that he knows the key to the trapdoor via some appropriate zero-knowledge protocol. But that brings us back to our original problem of sequential composition. However, the zero-knowledge property is not really important in Vic's case. It only suffices that Peggy does not learn anything about his trapdoor. Beyond that Vic does not really care if additional information is leaked to Peggy. This extra flexibility allows Vic to use the parallel version of the appropriate protocol to convince Peggy that he knows the trapdoor information. Such a protocol is called a *witness hiding protocol* in [FeS1] and while being a weaker requirement than zero-knowledge, it still satisfies the security requirements of most cryptographic protocols. (For more on witness hiding protocols see [FeS2])

Thus we can obtain a 2-round (4-move) perfect zero-knowledge protocol. In the first round Vic proves to Peggy that he knows the trapdoor information to the bit commitment scheme using the parallel version of the appropriate zero-knowledge protocol. If Peggy is convinced, she uses the parallel version of the actual protocol using Vic's trapdoor blobs

which takes another round. This achieves a 2-round perfectly zero-knowledge protocol for all languages in NP.

§5.4 The Complexity of Zero-knowledge Protocols

In the last two chapters we have reviewed two general methods for the construction of zero-knowledge protocols for languages L in NP. One involves the transformation of L to a graph 3-colourability instance and the other proceeds by reduction to a verifying Boolean circuit. We can now compare the communication cost of each method. Let L be a language which has a zero-knowledge protocol and let $CC_k(n)$ be the number of bits communicated during the course of this protocol where n is the size of the input and k is a security parameter selected such that the probability of error does not exceed 2^{-k}.

The first method involves proving that the resulting graph is 3-colourable so we let n be the number of vertices and e the number of edges in the resulting graph. If encryption cost is constant then the communication cost, CC_n^{3col}, incurred in proving that the graph is 3-colourable is such that CC_n^{3col} is $O(k \cdot e \cdot n)$. However we have to reduce the language L to a SAT instance before we can transform it to a graph 3-colourability instance. The size of the SAT instance depends on the time complexity of the non deterministic Turing machine (NDTM) for L. If the time complexity is linear in the size of the problem instance, we obtain a SAT instance of size $O(n^2)$ which gives a graph with $O(n^2)$ vertices and edges and hence $CC_k(n)$ is $O(k \cdot n^4)$. If the time complexity is quadratic in the size of n as it would be for many number theoretic languages, $CC_k(n)$ is then $O(k \cdot n^8)$. Clearly such a cost is prohibitive and cannot be used for practical purposes for arbitrary languages in NP.

In the circuit-based methods the cost $CC_k(S)$ is $O(k \cdot S)$ where S is the size of the verifying circuit. It has been shown that if the time required for a Turing machine to verify membership in a language L is $p(n)$ for a problem of size n then there exists a verifying

circuit of size $O(p(n) \cdot logp(n))$. Consequently, if $p(n)$ is linear then $CC_k(n)$ is $O(k \cdot nlogn)$ and if $p(n)$ is quadratic then $CC_k(n)$ is $O(k \cdot n^2 logn)$.

Thus the circuit-based method is the more suitable method for practical implementation. However, if the known techniques of truth tables are used the hidden constants in the expressions above turn out to be quite large. It is therefore important to find methods by which the number of bits communicated during the protocol be reduced to a minimum. In the next sections we outline two techniques for doing this, one due to Boyar and Peralta [BP] which eliminates the need for truth tables and the other due to Killian, Micali, and Ostrovsky [KMO] which reduces the number of blobs required.

§5.4.1 The Brassard-Crépeau Circuit-based Proof System [BC2]

To eliminate truth tables, Boyar and Peralta [BP] use techniques based on the original circuit-based methods devised by Brassard and Crépeau [BC2]. Since the latter are slightly different from their subsequent methods presented in §4.3.2, we give a brief outline of their original method.

In [BC2] the rows of each truth table are permuted as usual but there are no independent wire complementations. Instead for each wire in the circuit Peggy determines which value would be carried by that wire if the input wires were carrying the values corresponding to the satisfying assignment. Peggy then commits to the bits in all the truth tables as well as to the bits carried on the wires.

If Vic issues challenge A, Peggy opens the blobs corresponding to all the truth tables and Vic verifies that they correspond to valid permutations of the truth tables. If Vic issues challenge B Peggy shows that the inputs to and output of each gate correspond to some row in the truth table for that gate. This is done using the blob equality property of blobs where Peggy shows that the blobs on the circuit wires which are the inputs and the output of each gate correspond to the blobs in some row of the truth table.

§5.4.2 Eliminating Truth Tables

In this section we outline the techniques used in [BP] to eliminate the need for truth tables thereby reducing the number of bits communicated during the course of a zero-knowledge protocol.

MAJORITY Gates

A majority gate with fan-in $n = 2k + 1$ is such that the output is one if at least $k+1$ of the inputs are one, and zero if at least $k+1$ of the inputs are zero. To demonstrate, in a zero-knowledge fashion, that the gate is working it suffices to show that $k+1$ of the inputs encrypt the same bit as the output. In order to hide which of the inputs are the same as the output, Peggy produces n $(= 2k + 1)$ additional blobs which she can open to reveal the same bits as the inputs to the gate. However, she sends them to Vic in a random order so as to prevent him from learning the correspondence.

If Vic's challenge is A, Peggy shows the correspondence between the input blobs and the n additional blobs using the equality property. If Vic's challenge is B Peggy shows that $k+1$ of the additional blobs and the output blobs encrypt the same bit. It is clear the above protocol reveals nothing about the value of the inputs or the output. We next show how MAJORITY gates can be used to simulate AND/OR gates.

Simulating AND/OR Gates using MAJORITY Gates

Consider an AND gate with n inputs. This can be simulated by a MAJORITY gate with $2n$-1 inputs, n of which are the inputs to the AND gate whereas the remaining n-1 carry the value zero. Thus, the output of the MAJORITY gate will be one if and only if all the inputs to the AND gate were all one as required. OR gates can be simulated in the same manner except that the n-1 additional inputs to the MAJORITY gate must now all be one. In the latter case if at least one of the inputs to the OR gate is one the output of the MAJORITY gate will be one as required. We can now describe how majority gates can be used to simulate AND/OR gates.

For an AND/OR gate with n inputs, Peggy creates $2n$-1 additional blobs n of which correspond to the inputs of the AND/OR gate and the remaining n-1 corresponding to 0/1 for AND/OR. If Vic issues challenge A, Peggy shows the correspondence between the inputs to the AND/OR gate and n of the additional blobs and opens the remaining n-1 blobs to reveal a 0/1. If Vic issues challenge B, Peggy shows that n of the additional blobs encrypt the same bit as the output of the AND/OR gate. NOT gates require no additional blobs. If Vic issues challenge A, Peggy shows that the input blob and the output blobs encrypt different bits. If Vic issues challenge B, Peggy can ignore NOT gates.

We can now compare the truth table method with the MAJORITY gate method. If the AND/OR gates had 2 inputs, Peggy would have to create 12 blobs for the truth table and 1 more for the output blob. With the new method we only require $2n$-1 = 3 additional blobs, a substantial saving. If the number of inputs is N and considering a circuit with S gates, the total number of blobs on the circuits wires is N+S. If T of the gates are NOT gates, then the truth table based method uses $N + 13(S\text{-}T) + 5T$ where the second term represents the size of the truth tables for AND/OR gates and the third term the size of a truth table for NOT gates. On the other hand the new method only requires N +4(S-T) + T blobs per round which represent about a threefold increase in efficiency over the former method.

§5.4.3 Reducing the Number of Blobs

In the previous section we presented a technique for reducing the number of bits that need to be communicated during the course of a zero-knowledge protocol. Each of these bits must be encrypted before it is sent to the verifier and it is clear that to encrypt a message one must send more bits than the message itself. If the number of bits required to encrypt a bit is O(k) we say that the cost of the blob is O(k) for a security parameter k. For example, to build a blob based on a probabilistic encryption scheme one might have to use 100 bits. Thus to commit to 10000 bits individually we would need to send a million bits. However, we note that in zero-knowledge protocols, Peggy wants to commit to a sequence of bits

$b_1,b_2,..,b_m$ to be revealed subsequently *at the same time*. Thus instead of having one bit per blob, it would be more desirable to stuff all m bits into a larger blob which could then be opened to reveal all the desired bits. Killian, Micali, and Ostrovsky [KMO] have presented such a protocol for aggregate bit commitment such that when it is applied to the above example, it is possible to commit to 10000 bits in aggregate using only 11000 bits of communication. The protocol (called a subset revealing protocol by [KMO]) retains the zero-knowledge property and works as follows.

Aggregate Bit Commitment Protocol

To commit to a set of bits B = $b_1,b_2,..,b_m$, Peggy chooses a set of random bits R = $r_1,r_2,..,r_m$. She then commits to a blob containing R and another containing R $\oplus$ B (the bitwise XOR of B and R). To open the blob to reveal a subset I $\subseteq$ [1,n], the prover reveals I and for each $i \in$ I, she sends r_i and and $b_i \oplus r_i$. Vic then asks Peggy to open the blobs corresponding to either R or R $\oplus$ B and checks that the committed values correspond to the values sent by Peggy. If they are equal, Vic can now compute b_i for all $i \in$ I. An efficient protocol for committing to many bits has been implemented by Naor [N] using a pseudo random generator. Her main result is a protocol such that if m is at least linear in the security parameter k, Peggy can commit to B while exchanging only O(k) bits. The main theorem is as follows:

Theorem 4.4 [N]: If G is a pseudo-random generator then we can build an aggregate bit commitment scheme to B = $b_1,b_2,..,b_m$ such that for any polynomial p and large enough security parameter k :

1. After the commit stage, Vic cannot guess any bit b_i with probability greater than $\frac{1}{2} + \frac{1}{p(k)}$, even when told the values $b_1,b_2..b_{i-1},b_{i+1},..,b_m$.

2. For all $1 \leq i \leq m$, Peggy can only reveal 1 possible value for b_i except with probability less than $\frac{1}{p(k)}$.

This method of bit commitment greatly reduces the number of bits that are used for building blobs in zero-knowledge protocols.

§5.5 Non Interactive Zero Knowledge Proofs [BFM]

Three main characteristics of zero-knowledge protocols differentiate them from traditional ones.

1. Interaction. The prover and the verifier exchange messages back and forth.
2. Hidden randomization. The verifier generates random bits in a way that cannot be predicted by the prover.
3. Computational Difficulty. The prover imbeds in her proofs the computational difficulty of some other problem.

Although it appears that all these conditions are necessary, it is important to extract from them the strict minimum conditions required to preserve the zero-knowledge aspects of the proofs. Blum, Feldman, and Micali [BFM] introduced the concept of a non interactive zero-knowledge (NIZK) proof by eliminating the need for interaction (characteristic 1) . In doing so they have also eliminated the need for secrecy in generating the required random bits. Their main result states that a prover can prove in zero-knowledge and without any interaction with the verifier any statement T in NP provided that the prover and the verifier share a common random string σ. That is, Peggy gives Vic a string m and upon examining m, Vic is convinced that T is true but obtains no additional knowledge about the proof. Note that sharing a random string is a weaker assumption than interaction. Obviously, if Peggy and Vic could interact they could generate a random string while the converse is not true. Such protocols are attractive since random public sources are available such as the 1 000 000 random digits published by the RAND corporation.

The [BFM] implementation is based on the difficulty of distinguishing products of two primes from products of three primes. De Santis, Micali, and Persiano [DMP1] have based their implementation on the difficulty of distinguishing quadratic residues from non

residues. Under the assumption that oblivious transfer protocols exist, [KMO] and Bellare and Micali [BM] have shown that at the cost of an initial preprocessing stage, the prover can prove polynomially many statements in NP. However, these proofs are directed at a specific verifier and are not publicly verifiable. Under the more general assumption that one-way functions exist [DMP2] and Lapidot and Shamir [LS] have devised a model whereby the prover first proves a random theorem T_0 in an interactive preprocessing stage and uses it to prove the actual theorem T non-interactively. The main results of [LS] is a *publicly verifiable* NIZK proof for all languages in NP based on a common random string under the assumption that one-way permutations (a permutation that can be easily evaluated but cannot be inverted in polynomial time) exist. This eliminates the need for an initial interactive preprocessing stage. If the prover is restricted to polynomial-time computations the assumption needs to be strengthened to the effect that trapdoor permutations exist (a trapdoor is similar to a one-way permutation except that there is a secret known as the key to the trapdoor which allows one to easily invert the permutation), at the expense of needing a longer common random string. A drawback of the [LS] scheme is that it is *bounded* in that only a single theorem can be proven by a NIZK proof using a particular common random string. Independently, Feige, Lapidot, and Shamir [FLS] and De Santis and Yung [DY] have shown how to transform any bounded NIZK proof system with polynomial time provers into a more general NIZK system where polynomially independent provers can share the *same* random string which can then be used to prove polynomially many statements in NP.

It is clear that by eliminating interaction, the possibilities for practical uses of NIZK proof systems are much greater and in the next chapter we will present some of their cryptographic applications.

Chapter 6
Cryptographic Applications of Zero-Knowledge

§6.1 Introduction

Since their introduction in 1985 the application of zero-knowledge to cryptography has been a subject of continual research. In this chapter we will give a sample from the literature of some of the more important contributions of zero-knowledge protocols to cryptographic applications.

§6.2 Zero-Knowledge and Public Key Cryptosystems [GHY]

A public key cryptosystem is one in which each user, A, owns related keys P_A and S_A, where P_A is A's public key and S_A her secret key. In order to send a message m to A, another user B computes $c = E(P_A, m)$ and sends c to A who retrieves $m = D(S_A,c)$. E and D are polynomial-time algorithms and it is computationally infeasible to figure out m without the secret key S_A. In the next sections we discuss one application each of interactive zero-knowledge (IZK) and non-interactive zero-knowledge to public key cryptosystems (PKC).

§6.2.1 IZK and Public key Cryptosystems

Let $P_A = N = pq$ where p and q are large primes known to A. Using a zero-knowledge protocol A can convince any other user that $P_A = pq$ and that she knows the factors. Recall from §3.7.5 that

$$\text{Res}_N(x) = \begin{cases} 1 \text{ if } x \text{ is a quadratic residue mod } N \\ 0 \text{ otherwise} \end{cases}$$

If $x \in Z_N^*$ (+1) and A uses the result indistinguishable protocol of §3.9.4 to prove to B that she knows the value of $\text{Res}_N(x)$, x can be used as an encoding for bit $\text{Res}_N(x)$. Thus the sequence of random numbers $x_1,x_2,...x_k$ can be used as an encoding for the sequence of

bits $\mathrm{Res}_N(x_1)$, $\mathrm{Res}_N(x_2)$,...,$\mathrm{Res}_N(x_k)$ which can then be used as a one-time pad sent from A to B or vice-versa. A much more efficient system can be obtained if the sequence $\mathrm{Res}_N(x_1)$, $\mathrm{Res}_N(x_2)$,...,$\mathrm{Res}_N(x_k)$ is used instead as the random seed for a cryptographically secure pseudo-random bit generator based on P_A. Since A and B both share the seed both can use it to generate polynomially many bits which can be used as a very long one-time pad with which to send messages back and forth.

In most pubic key cryptosystems the use of public keys is asymmetric in that only messages sent to A can be encrypted using A's public key. In the system described above, the keys are symmetric, since messages sent to A as well as messages A sends to others are encrypted using A's public key. This can be useful in certain applications. For example it allows secure communication with casual users who are not registered in the public key directory. Also, since the same key is being used, A can transfer the same random bits to a group of users who can in turn broadcast secret messages to other members.

§6.2.2 NIZK, Public key Cryptosystems and Solving an Open Problem

Using NIZK proof systems has lead to the solution of a well-known open problem. In the chosen ciphertext attack, believed to be the strongest of all known natural attacks, the cryptanalyst attempts to break the system by asking and receiving as many plaintext (m) and ciphertext (c) messages of his choice. This would be feasible for any individual with access to the decoding equipment. It has been shown that Rabin's implementation which is based on the difficulty of factoring is easily vulnerable to such an attack and designing a public key cryptosystem that would be invulnerable to such an attack has been an open problem since 1978.

Using NIZK proofs this problem can finally be solved. The basic idea, due to Blum, Feldman, and Micali, is that instead of A sending only the ciphertext c, that she also sends along a string σ which is a NIZK proof that the sender knows the decoding of c. The decoding equipment outputs the message m if and only if the proof, σ, is convincing and

outputs nothing otherwise. Thus, being able to use the decoding equipment is of no use as it will only output the original message if one can prove that one knows what the decoding will be. More simply, the decoding equipment can only be used to output what is already known. Naor and Yung [NY] have built a PKC which is provably secure against chosen ciphertext attacks, given an underlying PKC that is secure against passive eavesdroppers and a NIZK proof system with common random string. By combining the results of [NY] and [LS] we obtain the first example of a PKC provably secure against chosen ciphertext attacks that is not based on the computational complexity of a specific problem but is instead based on the more general assumption that one-way functions exist.

§6.3 Zero-Knowledge and Identification Schemes

In this section we describe identification schemes based on zero-knowledge. Furthermore due to their simplicity some of these schemes are suited to microprocessor-based devices such as smart cards and personal computers. We also describe limitations and possible abuses of such systems and how these can be remedied. An identification scheme is one by which Peggy can prove to Vic that she is Peggy in such a way that prevents Vic from impersonating Peggy.

§6.3.1 The Fiat-Shamir Identification Scheme

This scheme due to Fiat and Shamir [FiS] is a combination of zero-knowledge protocols and identity-based schemes. It assumes the existence of a trusted center which issues smart cards to users after properly checking their physical identity. Beyond that, no further interaction is required with the center.

Initial Setup

Before the center becomes operational, it chooses and makes public a modulus $n = pq$ where p and q are secret primes known only to the center and a pseudo-random function f

which maps strings in the range 0,1,...,n-1. The function f should be such that its output should be polynomially indistinguishable from the output of a truly random function.

Issuing a Card

When an eligible user applies for a smart card, the center prepares a string I which contains the relevant information about the applicant such as name, social insurance number, physical description, security clearance, expiration date, limitations on validity, etc. Since this is the information that will be verified by the scheme it is vital that it be made as detailed as possible and to double check its correctness. The center then performs the following computations.

1. For small values of j,

$$v_j = f(I,j)$$

2. For k values of j for which $v_j \in \mathrm{QR}_n$,

$$v_j s_j^2 \equiv 1 \mod n$$

3. Issue a smart card containing I, the k s_j values and their indices. For convenience we assume that the first k indices $j = 1,2,...,k$ are used.

Verifying the validity of a card

The verification devices are identical standalone devices containing a microprocessor, a small memory and I/O interface and stores the modulus n and the function f. A smart card must then prove to the device that it knows the s_j values without giving away any information about them. When the card (prover) is inserted in the device (verifier) the following protocol is executed.

1. The prover sends the string I to the verifier.
2. The verifier generates $v_j = f(I,j)$ for $j = 1,2,...,k$.

For i =1 to t repeat

3. The prover picks a random $r_i \in Z_n^*$, computes

$$x_i \equiv r_i^2 \bmod n$$

and sends x_i to the verifier.

4. The verifier sends a random vector $\mathbf{e_i} = (e_{i1}, e_{i2}, \ldots, e_{ik})$ to the prover.

5. The prover replies with

$$y_i \equiv r_i \prod_{e_{ij}=1} s_j \bmod n$$

6. The verifier checks that the following condition holds

$$x_i \stackrel{?}{\equiv} y_i^2 \prod_{e_{ij}=1} v_j \bmod n$$

If all t rounds are successful, the prover's identity is verified and accepted by the device.

Theorem 5.1. If the prover and verifier follow the protocol, the verifier always accepts.

Proof [FiS]. By definition

$$y_i^2 \prod_{e_{ij}=1} v_j \equiv r_i^2 \prod_{e_{ij}=1} s_{ij}^2 v_j \bmod n$$

$$\equiv x_i \bmod n. \square$$

Theorem 5.2. If the prover does not know the square roots s_j, the verifier will be fooled with probability at most 2^{-kt}.

Proof: The prover could cheat by guessing the correct vector **e** in each round and sending the following

$$x_i \equiv r_i^2 \prod_{e_{ij}=1} v_j \bmod n \text{ and } y_i = r_i$$

which could pass the verifier's check as being correct.

The probability of guessing the correct vector is 2^{-k} for each iteration and 2^{-kt} for the whole protocol.$\square$

Theorem 5.3. For a fixed k and arbitrary t, the above protocol is zero-knowledge.

Proof. Intuitively, the protocol reveals no information about the s_j's, since the x_i's are random squares and each y_i contains an independent random variable which effectively masks the values of the s_j. Hence all messages from the prover to the verifier consist of random numbers with uniform probability distribution.❑

Security

To attain a 2^{-20} security level (1 in a million) one could simply choose $k = 5$ and $t = 4$. Such a scheme was implemented by [Kn] and allows the authentication of a 120 byte identification string in an average time of six seconds.

§6.3.2 The Feige-Fiat-Shamir Identification Scheme

In this scheme first put forward by Feige, Fiat, and Shamir [FFS] and subsequently improved by Micali and Shamir [MS], the center's role is significantly diminished as its only purpose is to publish the secret modulus n of the required form. Each individual then chooses k random integers $s_1, s_2, \ldots, s_k$ and for each s_j s/he calculates

$$v_j \equiv \pm s_j^{-2} \mod n$$

where the sign is chosen randomly. He then keeps the s_j's secret and publishes the v_j's with which the individual's name is associated. The identification protocols consists of the user proving that he knows the s_j's without revealing anything about their values.

§6.3.3 Aspects of the Fiat-Shamir Scheme not related to Zero-knowledge

The following observations were made by Desmedt, Goutier, and Bengio [DGB] in relation to the identifications schemes presented above. In the Fiat-Shamir protocol the applicant's physical description is used as part of the string I to be used as an input to some one-way function f. If every time Peggy's identity is verified, her special string I is

adequately tested, meaning that the verification of the physical description (which is assumed to be unique) is always done by the verifier with 100% accuracy, then the security of the protocol is not based on zero-knowledge. This is because her s_j's are of no use to some other party whose physical description will be tested to reveal some other I' which corresponds to different s_j'. The main security feature is therefore the fact that physical description is unique and tested adequately.

In the Feige-Fiat-Shamir scheme, however, the physical description of the applicant is no longer part of the scheme. If we assume that physical description is not unique or is not adequately tested some other frauds are still possible one of which we proceed to describe.

§6.3.4 The Mafia Fraud

This has been dubbed the "mafia" fraud because of a statement by Shamir in reference to his identification scheme to the effect that "I can go to a Mafia-owned store a million successive times and they still will not be able to misrepresent themselves as me". The fraud works as follows.

A customer A is buying groceries at a market whose owner B is a member of the Mafia. At the same time another member C of the same gang is negotiating the purchase of diamonds in a jewelry store owned by D. C is linked to B via a secret radio link and C's identification card is linked via a full duplex radio channel to B's verification device. At the moment A is ready to pay, B informs his gang partner C of this fact. At this point C makes his choice of diamonds and D proceeds to check C's identity. However, C's card is linked to B's verification device and the jeweler is in fact verifying A's identity who gets stuck with a bill for diamonds he never purchased.

§6.3.5 The Subliminal Channel

We now show how the verifier can communicate information in a subliminal way to either the prover or an eavesdropper. A subliminal protocol is one in which it is possible to

"hide" another protocol. For example, in the scheme presented above the verifier can communicate a message M in a subliminal way to the smart card holder or to an eavesdropper as follows. Instead of choosing the challenge vectors e randomly, he lets them correspond to the encrypted message $E_k(M)$ using a secret or public key system. When the verifier issues the challenge a secret message is thus transmitted to the card holder. If the message is , instead, directed at an eavesdropper, the prover will still not be able to predict the bits corresponding to e and hence she is tested with the same degree of accuracy. Thus if banks were to use the Fiat-Shamir protocol, a dishonest clerk could relay confidential information in a completely undetectable way to eavesdropping members every time a customer uses her card . The discussion on subliminal channels is outside the scope of this work and interested readers are referred to [DBG].

§6.4 Improving the Fiat-Shamir Scheme

In the previous sections we presented some of the frauds possible with the Fiat-Shamir identification scheme; thus, such schemes although appearing secure on the surface should be used with caution. As we have seen the [FiS] scheme is such that the probability of undetected cheating is 2^{-kt} where k refers to the number of secret square roots stored and t is the number of rounds of the protocol. For example to attain a security level of 2^{-20} (tk = 20), when we reduce the number of interactions to $t = 1$ we must store $k = 20$ secret integers. Conversely, if we reduce the number of stored integers to $k = 1$ we must increase the number of interactions to $t = 20$. Thus there is a tradeoff between number of transmissions and memory size so that the efficient parameter values $t = k = 1$ cannot be used. This problem has been addressed in [GQ1,GQ2 and OO]. We present the scheme due to [GQ1,GQ2].

§6.4.1 The Guillou-Quisquater Authentication Scheme [GQ1]

This scheme achieves the optimal parameters $t = k = 1$ at the cost of longer computations. As before the center owns a public composite modulus n whose factors are kept secret. For each smart card with identity I the center produces some J such that $J = F(I)$ where F is a redundancy function .The value J produced by F, called the *shadowed identity* of the device, is a number as large as n. Half of its bits consist of the claimed identity I while the remaining half is completed by a redundancy which depends on the value of I. Each device then holds an authentication number B such that

$$B^v J \equiv 1 \bmod n$$

where v is a public exponent. Note that it is important that the factors of n as well as v be chosen carefully to ensure that B can always be extracted from J.

Whenever the card is inserted in the verification device, the following protocol is executed *exactly once*.

1. The card chooses a random integer $r \in Z_n^*$ and computes

   $$T \equiv r^v \bmod n.$$

   and sends T to the verifier.

2. The verifier picks a random value $d \in \{0,1,...v - 1\}$ and sends d to the prover as his challenge.

3. The prover computes a witness t as follows

   $$t \equiv r B^d \bmod n.$$

4. The verifier checks that the following condition holds

   $$J^d t^v \stackrel{?}{\equiv} T \bmod n$$

   If the above condition holds then the card is validated.

Note that if both parties follow the protocol then

$$J^d t^v \equiv J^d (r B^d)^v \bmod n$$

$$\equiv \quad (J\,B^v)^d\, r^v \bmod n$$

$$\equiv \quad r^v \bmod n \ \text{ since } J\,B^v \equiv 1 \bmod n.$$

A device knowing the authentication number B can easily answer any challenge. If the cheating prover is able to guess in advance the question d then she could easily prepare the set of integers t and T that would pass the verifier's test. [GQ1] show that knowing two witnesses t_1 and t_2 corresponding to two different challenges d_1 and d_2 for the same test number T gives significant (and generally total) knowledge about the authentication number B. Thus any cheater is able to produce in advance at most one witness number and by guessing the correct challenge d has probability at most $\frac{1}{v}$ of fooling the verifier. Thus by fixing the size of v to achieve the desired level of security the need for several iterations of the protocol is eliminated. Using Ali Baba's cave as an analogy, this would correspond to a cave with v passages as opposed to 2.

§6.4.2 Cooperation between Devices [GQ2]

There are 2 variations of this scenario. In the first case users with different identities can cooperate in a way that will make them look like a new user. In the second case the secret is partitioned among distinct devices sharing the same identity. In both cases cooperating parties reveal no information about their secret authentication numbers B.

§6.4.2.1 Same Exponent, Different Identities [GQ2]

Consider 2 security devices, each storing its unique authentication number B_1 and B_2 related to their identities I_1 and I_2 in the following way

$$B_1^v J_1 \equiv 1 \bmod n \qquad \text{where } J_1 = F(I_1)$$

$$B_2^v J_2 \equiv 1 \bmod n \qquad \text{where } J_2 = F(I_2)$$

The 2 devices now cooperate via a personal computer and negotiate an authentication transaction with the verifier according to the following protocol.

1a. Device I_1 chooses a random $r_1 \in Z_n^*$ and computes

$$T_1 \equiv r_1^v \mod n.$$

and sends T_1 and I_1 to the PC.

1b. Device I_2 chooses a random $r_2 \in Z_n^*$ and computes

$$T_2 \equiv r_2^v \mod n.$$

and sends T_1 and I_2 to the PC.

2. The PC computes a common test number T as

$$\begin{aligned} T &\equiv T_1 T_2 && \mod n \\ &\equiv (r_1 r_2)^v && \mod n \\ &\equiv r^v \mod n && \text{where } r = r_1 r_2 \end{aligned}$$

and sends I_1, I_2 and T to the verifier.

3. The verifier picks a random value $d \in \{0,1,...v - 1\}$ and sends d to both devices as his challenge.

4a. Device I_1 computes a witness t_1 as

$$t_1 \equiv r_1 B_1^d \mod n$$

and sends I_1 and t_1 to the PC.

4b. Device I_2 computes a witness t_2 as

$$t_2 \equiv r_2 B_2^d \mod n$$

and sends I_2 and t_2 to the PC.

5. The PC computes a common witness number t as follows

$$\begin{aligned} t &\equiv t_1 t_2 && \mod n \\ &\equiv (r_1 B_1^d)(r_2 B_2^d) && \mod n \\ &\equiv (r_1 r_2)(B_1 B_2)^d && \mod n. \end{aligned}$$

and sends t to the verifier.

6. The verifier checks that the following condition holds

$$J_1^d J_2^d t^v \stackrel{?}{\equiv} T$$

Note that if all parties carry out their share of the protocol then

$$J_1^d J_2^d t^v \equiv J_1^d J_2^d [(r_1 B_1^d)(r_2 B_2^d)]^v$$

$$\equiv (J_1 B_1^v)^d (J_2 B_2^v)^d (r_1 r_2)^v \bmod n$$

$$\equiv r^v \equiv T \bmod n \text{ as required.}$$

Note that this protocol can easily be extended to include any number of cooperating users.

§6.4.2.2 Same Identity, Different Exponents [GQ2]

In this case each of the two devices stores its unique authentication number B_1 and B_2 such that *both* are related to the common identity I as

$$J B_1^{v_1} \equiv 1 \bmod n \text{ and } J B_2^{v_2} \equiv 1 \bmod n \qquad \text{where } J = F(I)$$

The protocol allows two cooperating devices to simulate another device with entity I and exponent $v = v_1 v_2$ (assuming that v_1 and v_2 are relatively prime) where

$$B^v J \equiv 1 \bmod n \qquad \text{where } B_1 \equiv B^{v_2} \bmod n \text{ and } B_2 \equiv B^{v_1} \bmod n$$

As previously the two devices are cooperating via a PC and negotiate an authentication transaction with the verifier according to the following protocol.

1a. Device 1 picks a random $r_1 \in Z_n^*$ and computes

$$T_1 \equiv r_1^{v_1} \bmod n$$

and sends T_1 and I to the PC.

1a. Device 2 picks a random $r_2 \in Z_n^*$ and computes

$$T_2 \equiv r_2^{v2} \qquad \text{mod } n$$

and sends T_2 and I to the PC.

2. The PC computes a common test number T as

$$T \equiv T_1^{v2}\, T_2^{v1} \qquad \text{mod } n$$

$$\equiv (r_1\, r_2)^{v1\, v2} \text{ mod } n$$

$$\equiv (r_1\, r_2)^{v} \qquad \text{mod } n$$

$$\equiv r^{v} \qquad \text{mod } n$$

and sends T and I to the verifier.

3. The verifier picks a random value $d \in \{0,1,...v - 1\}$ and sends d to the PC who computes two questions d_1 and d_2 such that

$$d_1 \equiv d/v_2 \text{ mod } v_1 \text{ and } d_2 \equiv d/v_1 \text{ mod } v_2$$

which are sent to devices 1 and 2 respectively.

4a. Device I_1 computes a witness t_1 as

$$t_1 \equiv r_1\, B_1^{d_1} \qquad \text{mod } n$$

and sends I and t_1 to the PC.

4b. Device I_2 computes a witness t_2 as

$$t_2 \equiv r_2\, B_2^{d_1} \qquad \text{mod } n$$

and sends I and t_2 to the PC.

The PC computes the common witness t as

$$t \equiv rB^{d_1 v_2 + d_2 v_1} \qquad \text{mod } n.$$

5. The verifier then checks that the following condition holds

$$J^{d_1 v_2 + d_2 v_1}\, t^{v} \stackrel{?}{\equiv} T \qquad \text{mod } n$$

If the above condition holds the transaction is validated.

It is clear that the protocol can be extended to include arbitrarily many users. Note that cooperation protocols can be used to solve the problem of subliminal channels. It suffices

to have one of the cooperating devices act as a trusted warden whose purpose is to act as a go-between. This effectively destroys any messages that might be transmitted from the verifier to the prover.

§6.4.3 NIZK and Identification Schemes

In this setting, proposed by Bellare and Goldwasser [BG], a central authority needs to generate unique unforgeable ID's for its users. Any user should be able to present their ID in numerous, geographically distributed local stations which should be able to validate the ID's. Let $F_k = \{ f_s : |s| = k \}$, for some security parameter k, denote a collection of pseudo-random functions such that no probabilistic polynomial-time algorithm can distinguish a member f_s from a truly random function. A previous non-interactive ID scheme has been to select a random index s and to use the user's name and other related information as input to the function f_s to create the ID. A drawback of such a system is that all remote stations need to store and keep secret the index s to the random function. We can get around this difficulty by using the following scheme based on NIZK proofs.

In this scheme only the center possesses the secret index s to the pseudo-random function. The center picks a random value r and computes $\alpha = E(r,s)$ where E is a public encryption function, and publishes in a public file the pair (E,α). When a user U applies for an ID, the center computes $I = f_s(U)$ and issues a card to U containing I as well as a NIZK proof of the following statement

$$T = \exists s \exists r [\alpha = E(r,s) \text{ and } I = f_s(U)]$$

denoted $NIZK_\rho(T)$ where ρ is the common random string used to generate the NIZK proof. In this manner the local centers no longer need to store any special information. Whenever a user U wishes to authenticate himself he simply shows the center I and the NIZK proof of the above statement which convinces the center that the user possesses a legal ID number.

§6.5 Signature Schemes

A signature scheme is one by which Peggy can prove to Vic that she is Peggy but Vic cannot prove even to himself that he is Peggy. The difference between identification and signature schemes can be seen when we are considering interactive protocols. In identification schemes Vic could create a credible transcript by carefully selecting both questions and answers in the dialog whereas in signature schemes only real communication with Peggy could generate a credible transcript. We can illustrate this by considering the Mafia fraud described above. When we were using an identification scheme it was possible for a gang member to pass himself off as another user in a real-time fraud. A signature scheme solves this problem by linking user identity and transaction purpose in a unique transaction. The fraud then no longer works as two distinct transactions are being negotiated.

§6.5.1 The Fiat-Shamir Signature Scheme [FiS]

In the identification scheme of §6.3.1 the role of the verifier was to send vectors $\mathbf{e_i} = (e_{i1},e_{i2},...,e_{ik})$ whose unpredictability prevented the prover from cheating. By replacing the random vectors by a pseudo-random function f we can transform the identification scheme into a signature scheme by which Peggy signs a message m .

Protocol for Peggy's signing a message m

1a. Peggy picks random $r_1,r_2,...,r_t \in \{0,1,...,n-1\}$ and computes

$$x_i \equiv r_i^2 \bmod n \qquad (i= 1,2...,t)$$

1b. She then computes $f(m,x_1,x_2,...,x_t)$ and uses the first kt bits as the $\mathbf{e_i}$ vectors to compute

$$y_i = r_i \prod_{e_{ij}=1} s_j \quad \bmod n \text{ for } i= 1 \text{ to } t.$$

and sends I, m, the $\mathbf{e_i}$ vectors and the y_i's to the verifier.

Protocol for Vic to verify A's signature on m.

2a. Vic computes $v_j = f(I,j)$ for $j = 1$ to k and

$$z_i = y_i^2 \prod_{e_{ij}=1} v_j \qquad \text{mod } n \text{ for } i = 1 \text{ to } t$$

2b. He then verifies that the first kt bits of $f(m,z_1,z_2,...,z_k)$ correspond to the e_i vectors and if so, accepts the signature as valid.

Although the above scheme is not zero-knowledge, [FiS] show that the information about the secret square roots obtained by Vic from various signatures is not enough to be of any use in forging new signatures.

§6.5.2 NIZK and Signature Schemes [BG]

We now present a signature scheme using non-interactive zero-knowledge protocols. We use the same notation as §6.4.3. Consider Peggy's attempt to sign a message m. If ρ is a public random string, then for some security parameter k, a randomly chosen index s to the family of pseudo-random functions F_k and $\alpha = E(r,s)$, Peggy's public file PK_{Peggy} is then $\{E,\alpha,\rho)$ and her secret file is $\{r,s\}$.

Protocol to sign m

1. Peggy computes $c = f_s(m)$.
2. Peggy uses the common random string to produce a NIZK proof, $NIZK_\rho(T)$ where T is the following statement

$$T = \exists s \exists r[\alpha = E(r,s) \text{ and } c = f_s(m)]$$

3. Peggy sends c, m and $NIZK_\rho(T)$ to the verifier as the signature of m.

In this case anyone who has access to Peggy's public file can verify that the signature is indeed valid by checking the validity of the NIZK proof. If the proof is valid then the verifier is convinced that the signature was indeed produced by Peggy.

§6.6 Miscellaneous Notes

Other applications of zero-knowledge are given in the literature. For example Chaum and van Antwerpen [CV] have introduced the notion of *undeniable* signatures which unlike digital signatures can only be verified with the signer's cooperation. Other areas of application have been proposed such as in the implementation of electronic currency [CFN]. In the above sections some of the more significant and practical applications of zero-knowledge to cryptography have been presented.

Conclusion

In this thesis we have attempted to describe the principles of interactive proofs and zero-knowledge as well as give a broad view of the more important results arising from their study. Originally intended to solve a specific problem, interactive proofs have spilled over to other fields and given rise to new ones such as interactive complexity.

Interactive proofs are an extension of the classical NP notion of efficient provability, where non-determinism is enhanced by introducing two new features: randomness (the verifier is allowed to toss coins) and interaction (the prover and the verifier exchange a polynomial number of messages). The added power conferred by these new features has been demonstrated: for example, the graph non-isomorphism problem, although not known to be in NP nonetheless possesses an interactive proof system.

Zero-knowledge proofs have proved to be useful both in cryptography and complexity theory. On a theoretical level, results of Fortnow [F] have provided an avenue for deciding whether or not certain languages belong in NP. For example, one way to show that a language L (for which no efficient algorithm is known) in not NP-complete is to exhibit a perfect zero-knowledge proof for it. In cryptography, we have seen some of the numerous possible applications of zero-knowledge. However, most of the results obtained have required either unproven complexity assumptions or an unbounded number of exchanged messages. For example, the assumption that one-way functions exist, although a weak one, has been used extensively. From a cryptographic point of view most of the currently known one-way functions are almost exclusively based on number theory such as integer factorization and discrete logarithms. If these were found to be efficiently solvable the consequences would be disastrous for many of the results presented throughout this thesis. Consequently is it important that alternate models be developed where intractability assumptions are eliminated. One model which achieves this result was proposed by Ben-Or, Goldwasser, Killian, and Widgerson [BGKW1] which we now describe briefly.

Their multi-prover interactive proof system is an extension of the one-prover interactive proof system. Instead of one prover attempting to convince the verifier that $x \in L$, the prover is replaced by two separate provers who jointly agree on a strategy by which to convince the verifier that the statement is true. The provers are allowed to agree on a strategy before the start of the protocol with the verifier but once this interaction starts the provers can no longer talk to each other nor can they see the messages exchanged between the verifier and the other prover. Using this model, Ben-Or *et al.* [BGKW2] have shown perfect zero-knowledge protocols for all languages in NP without using any intractability assumptions. In their protocol the two provers share a common random string. The responsibility for proving the assertion rests mainly with one of the provers while the main role of the other prover is limited to periodically outputting portions of the random string he shares with the other prover. In effect, the verifier checks his communication with the provers by playing them against each other. This means that if one of the provers is cheating the verifier will catch her by asking the other prover what the valid response should have been. By repeating this process randomly, the verifier ensures that the provers cannot cheat. Note that in order for the verifier to believe the validity of the proof, it is his responsibility to ensure that the provers cannot communicate with each other while the proof is taking place. However, even if this condition does not hold the interaction is still zero-knowledge. In a cryptographic setting one can consider the verifier to be a bank which issues two cards (the provers) to its users. Even more interestingly, the interactive proofs based on the two-prover model remain zero-knowledge even when the rounds are executed in parallel.

As we have seen in Chapter 4, the security of zero-knowledge protocols relied on the security of the bit commitment schemes being used and these required typically complex operations to implement. Using the two-prover model bit commitment can be considerably simplified and we next describe how this is done.

Let the two provers be denoted P_1 and P_2 and the verifier as V. In this model P_1 and P_2 share a common random string CRS = $r_0,...,r_k$ where $r_i \in \{0, 1, 2\}$ and k is the number of bits that can be committed to using the CRS which we assume is polynomially bounded. The following functions are known to all parties involved. Let σ_0 denote the identity function, $\sigma_1(0) = 0$, $\sigma_1(1) = 2$ and $\sigma_1(2) = 1$. Let r_j denote the jth bit of the CRS. In order to commit to some bit b where b is the jth bit being committed to, P_1 and V execute the following protocol

1. V generates a random bit $c \in \{0, 1\}$ and sends c to P_1.
2. P_1 computes $v_j \equiv \sigma_c(r_j) + b \bmod 3$ and sends v_j to V.
3. V stores (j, c, v_j).

In order to reveal the *jth* bit b the following is executed

1. P_2 sends r_j to V who can then compute b as

$$b \equiv v_j - \sigma_c(r_j) \bmod 3.$$

Using the above bit commitment scheme Ben-Or *et al.*[BGKW2] have built an identification scheme that is considerably more computationally efficient that the ones based on the one prover model since the complex operations required for bit commitment are now replaced by simple additions modulo 3.

Several aspects of interactive proofs, both theoretical and practical, remain to be investigated. The introduction of non-interactive zero-knowledge proofs has eliminated the need for interaction as well as the need for secrecy of the random bits exchanged between prover and verifier. Although randomness appears to be an essential feature of interactive proofs, only recently have researchers turned their attention to the number of random bits required during the course of such proofs (e.g. [BGG]), and much work remains to be done. In the one-prover model, Bellare, Micali and Ostrovsky [BMO] have found constant-round perfectly zero-knowledge protocols for graph isomorphism and quadratic residuosity *which do not rely on complexity assumptions*. It would be interesting to find

such protocols for other languages which ultimately lead to a class of such languages whose structure could then be studied. Also, as we have seen, different models for interactive and zero-knowledge protocols have been proposed. It would be desirable to classify these different models in terms of to which cryptographic applications each model is best suited. In view of the increasing proliferation of networks, the secure transmission of information across possibly insecure channels will likely become a major problem. As we have seen, zero-knowledge and witness-indistinguishable protocols can help alleviate these problems. It would therefore seem important that these techniques be implemented either as standalone schemes or that they be incorporated into existing schemes. With the implementation of such systems the passage of zero-knowledge to the mainstream of cryptography will be complete.

Appendix A
Cryptographic Capsules

Cryptographic capsules, a tool invented by Cohen [Co] have proven to be quite useful in zero-knowledge protocols. Informally, a cryptographic capsule is a randomly ordered collection of objects each of which is of a specified form. The purpose of the random ordering is to hide the form of the various objects involved. The main feature of capsules is that they allow a prover to convince a verifier, in an interactive manner that a capsule is of a specified form without revealing anything about its contents.

In their application most relevant to this thesis, [Co] has shown that it is possible to demonstrate that two integers belong to the same residue class without revealing any information about which class it is. We will need the following definitions.

Definition: Given integers n and y, y is said to be an rth residue mod n if there exists some integer x such that

$$y \equiv x^r \bmod n.$$

We can now define residue classes.

Theorem 1[Co]: Let $\phi(n)$ denote Euler's totient function. Let n and r be defined such that $r|\phi(n)$ and $r^2 \nmid \phi(n)$. If $(y,n) = 1$ and y is not an r^{th} residue mod n, then for every w such that $(w,n) = 1$, w can be expressed as

$$w \equiv x^r y^i \bmod n,\ 0 \le i < r$$

for some integer x.

We call i the residue class index of w with respect to n, y, and r. All elements which have the same residue class index w.r.t. n, y, and r are said to belong to the same residue class.

Proof. A proof of the above statement does not appear in [Co] and we will first show that the theorem does not hold in general. Indeed, consider the following counter-example to Theorem 1.

Let $Z_n^* = \{x \mid (x,n) = 1 \text{ and } 1 \le x \le n-1\}$. Z_n^* forms a group under the operation of multiplication modulo n. Let $r|\phi(n)$ where $\phi(n)$ is the order of the group.
Consider now the set $H = \{x^r \mid x \in Z_n^*\}$. Clearly, H is a subgroup of Z_n^*.

If $w \in Z_n^*$, then $w \in Hg$ where $g \in Z_n^*$.. Thus if $w \equiv x^r y^i \bmod n$ then $w \in Hy^i$. We will show that this is false for the case where $n = 77$ and $r = 15$ and $\phi(n) = 6 \times 10 = 60$.

In building the set H is suffices to consider those elements from 1 to n that are relatively prime to n as follows

C = {1,2,3,4,5,6,8,9,10,12,13,15,16,17,18,19,20,22,23,24,25,26,27,29,30,31,32, 34,36,37,38,...}.

We build H by raising the above elements of C to the power of $r = 15 \bmod n = 77$. If we do this for all elements of C, we find that only the following four elements are in H

H = {1,43,34,76}.

We now select some y such that y is not an r^{th} residue i.e. $y \notin H$. Let $y = 65$ and build the cosets Hy^i for different values of i.

Case $i = 0$. H = {1,43,34,76}

Case $i = 1$. H65 = {65,23,54,12}

Case $i = 2$. $H65^2$ = H67 since $65^2 \equiv 67 \bmod 77$
= {67,32,45,10}

Case $i = 3$. $H65^3$ = H43 since $65^3 \equiv 43 \bmod 77$
= {43,1,76,34} = H

Therefore $H' = H \cup H65 \cup H67$ are the only elements that can be expressed as $w \equiv x^{15}65^i \bmod 77$. If we now consider the integer $5 \notin H'$ and $(5,77) = 1$, we find that $5 \notin H'$ and hence cannot be expressed as $x^{15}65^i$ which disproves the theorem.❑

As mentioned before we now reformulate Theorem 1 to read as follows.

Theorem 1* : Let $\phi(n)$ denote Euler's totient function. Let n and r be defined such that $r|\phi(n)$, $r^2 \nmid \phi(n)$ and r *is prime*. If $(y,n) = 1$ and y is not an r^{th} residue mod n, then for every w such that $(w,n) = 1$, w can be expressed as

$$w \equiv x^r y^i \bmod n,\ 0 \le i < r$$

for some integer x.

Before proving the above theorem, we will need the following lemma.

Lemma 1. Given the sets Z_n^* and $H = \{x^r \mid x \in Z_n^*\}$, we will show that $|H| = \frac{\phi(n)}{r}$.

Proof. Let $x^r \equiv y^r \bmod n$ where $x \not\equiv y$. By Sylow's Theorem, there exists w such that $w^r \equiv 1 \bmod n$, $w \equiv xy^{-1} \bmod n$ and $w \not\equiv 1 \bmod n$.
Then the set $\Omega = \{1, w, w^2, \ldots, w^{r-1}\}$ is a subgroup of Z_n^* of order r. Let $S \subset Z_n^*$ and the group Z_n^* can be expressed as a sum of cosets

$$Z_n^* = \bigcup_{g \in S} \Omega g$$

If we define $\Omega_g^{(r)} = \{x^r \mid x \in \Omega g\}$ then H can be expressed as

$$H = \bigcup_{g \in S} \Omega_g^{(r)}$$

Now, if $x \in \Omega g$ then

$$x \equiv w^i g \bmod n$$

$$x^r \equiv w^{ir} g^r \bmod n$$

$$\equiv g^r \bmod n \text{ since } w^r \equiv 1 \bmod n$$

and

$$H = \bigcup_{g \in S} \{g^r\}$$

so that $|H| = |S| = \frac{\phi(n)}{r}$.❑

We can now go on to prove the main theorem.

Proof (Theorem 1*). Let $\gamma \in Z_n^*$ so that $\gamma \notin H$ (i.e. γ is not an rth residue mod n). It is obvious that $\gamma^r \in H$. Let t be the least positive integer such that

$\gamma^t \in H$ and let $r = qt + s$, $0 \leq s < t$.

Now, $\gamma^{qt+s} \in H$,

but $\gamma^t \in H \Rightarrow \gamma^{qt} \in H$,

hence, $\gamma^s \gamma^{qt} \in H \Rightarrow \gamma^s \in H$ which is a contradiction since our initial condition is that t is the minimal value for which the above holds. Note that $s = 0 \Rightarrow t \mid r \Rightarrow t = r$, since r is prime.

Consider now the following distinct cosets

$H, H\gamma, H\gamma^2, \ldots, H\gamma^{r-1}$.

There are r such cosets each containing $\frac{\phi(n)}{r}$ (lemma 1) for a total of $\phi(n)$ elements which is the exact order of Z_n^*. Thus any $w \in Z_n^*$ can be expressed as

$$w \equiv x^r \gamma^i \bmod n \ .\square$$

The following lemma on cryptographic capsules due to [Co] then holds.

Lemma 2. Two integers x_1 and x_2 such that $(x_1,n) = (x_2,n) = 1$ belong to the same residue class with respect to n, y, and r iff there exists some integer v such that

$$v^r \equiv \frac{x_1}{x_2} \bmod n.$$

Thus to show that 2 integers are of the same residue class, one only needs to exhibit an r^{th} root of their quotient.

Appendix B

In this appendix we will define Legendre and Jacobi symbols and give some of their well-known properties. We will then use those to prove some important results as promised in §3.9.3.

Legendre and Jacobi Symbols

Consider a prime $p > 2$ and an integer a such that $p \nmid a$. Then, by Fermat's theorem $a^{p-1} \equiv 1 \bmod p$. Thus,

$$(a^{(p-1)/2}-1)(a^{(p-1)/2}+1) \equiv 0 \bmod p.$$

If $a \in QR_p$, then $p \mid a^{(p-1)/2}-1$ by the following argument. Since $a \in QR_p$, there exists an integer x, such that $x^2 \equiv a \bmod p$. Therefore,

$$\begin{aligned} a^{(p-1)/2} \bmod p &\equiv [x^2]^{(p-1)/2} \bmod p \\ &\equiv x^{p-1} \bmod p \\ &\equiv 1 \bmod p \end{aligned}$$

and $a \notin QR_p$ if $p \mid a^{(p-1)/2}+1$.

When p is prime we have three possibilities.

1. $p \mid a$.
2. $a^{(p-1)/2} \equiv 1 \bmod p \Rightarrow a \in QR_p$.
3. $a^{(p-1)/2} \equiv -1 \bmod p \Rightarrow a \notin QR_p$.

We define the Legendre symbol $\left[\frac{a}{p}\right]$ as follows:

1. $\left[\frac{a}{p}\right] = 0$ if $p \mid a$.
2. $\left[\frac{a}{p}\right] = 1$ if $a \in QR_p$.
3. $\left[\frac{a}{p}\right] = -1$ if $a \notin QR_p$.

We see that $\left[\frac{a}{p}\right] = a^{(p-1)/2} \bmod p$.

We will now list some well-known properties of Legendre symbols.

1. $\left[\frac{a}{p}\right] = \left[\frac{b}{p}\right]$ if $a \equiv b \bmod p$.

2. $\left[\frac{ab}{p}\right] = \left[\frac{a}{p}\right]\left[\frac{b}{p}\right]$

3. $\left[\frac{1}{p}\right] = 1$

4. $\left[\frac{2}{p}\right] = \left.\begin{array}{l} 1 \text{ if } p \equiv \pm 1 \bmod 8 \\ -1 \text{ if } p \equiv \pm 3 \bmod 8 \end{array}\right\}$

5. $\left[\frac{p}{q}\right]\left[\frac{q}{p}\right] = -1^{(p-1)/2 \times (q-1)/2}$ where p,q are prime.

The last property is known as the law of quadratic reciprocity.

The Jacobi symbol of some integer $a \bmod N$ of the form $N = \prod_{i=1}^{k} p_i^{e_i}$ (for p_i prime) is defined as

$$\left(\frac{y}{N}\right) = \prod_{i=1}^{k}\left[\frac{y}{p_i}\right]^{e_i}$$

where $\left[\frac{y}{p_i}\right]$ is the corresponding Legendre symbol.

For any equation $a \equiv x^2 \bmod n = pq$ (p,q prime) it can be shown that a has 4 roots given by x, $n - x$, y, $n - y$ [De]. We can now prove the following lemma.

Lemma. Given n,x,y and a as defined above, we show that GCD$(x+y,n) = p$ or q, where GCD refers to Greatest Common Divisor.

Proof. Since x and y are roots we have

$$x^2 \equiv y^2 \equiv a \bmod n.$$

Then $x^2 \equiv y^2 \bmod p \Leftrightarrow p \mid x^2 - y^2 \Leftrightarrow p \mid x - y$ or $p \mid x + y$.

and $x^2 \equiv y^2 \bmod q \Leftrightarrow q \mid x^2 - y^2 \Leftrightarrow q \mid x - y$ or $q \mid x + y$.

Case 1. If $p \mid x - y$ and $q \mid x - y$, $pq = n \mid x - y$.

But $0 < x,y < n \Rightarrow -n < x - y < n$, therefore

$x - y = 0$ or $x = y$.

Case 2. If $p \mid x + y$ and $q \mid x + y$, $pq = n \mid x + y$.

But $0 < x,y < n \Rightarrow 0 < x - y < 2n$, therefore

$x + y = n$ or $x = n - y$.

Case 3. If $p|x - y$, $p \nmid x + y$, and $q|x + y$, $q \nmid x - y$. (Without loss of generality or else we simply exchange p and q). Then, $q|x + y$ and $q|n \Rightarrow q|\text{GCD}(x + y,n)$.

Let $d = \text{GCD}(x + y,n)$ and let b be such that $d = bq$. Then $b|d|(n = pq)$. We identify the following subcases.

1. $b = n$. Then, $p|b|d|x + y$, which is a contradiction.

2. $b = p$. As above by the same argument.

3. $b = q$. Then, $bq|n = pq \Rightarrow p = q$ which is contradiction.

4. $b = 1$. Then, $q = d = \text{GCD}(x + y, n)$.

By a similar argument we can show that $p = \text{GCD}(x - y, n)$.❑

Theorem. Given x, y, a, and $n = pq$ as defined above, where $p \equiv q \equiv 3 \bmod 4$, if one can extract 2 square roots of $a \equiv x^2 \bmod n$ with different Jacobi symbols, then one can factor n.

Proof. As mentioned before, a has 4 square roots x, $n - x$, y, $n - y$. Let the Jacobi symbol of x, $\left(\frac{x}{n}\right) = s$ where $s \in \{1, -1\}$. Then,

$$\left(\frac{n - x}{n}\right) = \left(\frac{-x}{n}\right)$$

$$= \left(\frac{-1}{n}\right)\left(\frac{x}{n}\right) \quad \text{(By property 2 of Legendre symbols)}$$

$$= \left[\frac{-1}{p}\right]\left[\frac{-1}{q}\right]\left(\frac{x}{n}\right)$$

$$= -1 \times -1 \times s \quad \text{since } p \equiv q \equiv 3 \bmod 4 \text{ where}$$

$$\left[\frac{-1}{p}\right] = -1^{(p-1)/2} \bmod p = -1 \text{ and}$$

$$\left[\frac{-1}{q}\right] = -1^{(p-1)/2} \bmod q = -1.$$

$$= s.$$

Therefore, x and $n - x$ have the same Jacobi symbol. But we know that half of the roots of a must have Jacobi symbol $-s$. Hence, y and $n - y$ must have precisely Jacobi symbol $-s$. But from the above lemma, knowing one of $\{x, n - x\}$ (Jacobi symbol s) and one of $\{y, n - y\}$ (Jacobi symbol $-s$) enables one to factor n by obtaining one of p or $q = \text{GCD}(x + y, n)$. Therefore, obtaining two square roots with different Jacobi symbols enables one to factor n as stated.❑

References

[AABFH] M. Abadi, E. Allender, A. Broder, J. Feigenbaum, and L.A. Hemachandra, "On generating solved instances of computational problems", in Advances in Cryptology: Proceedings CRYPTO 88, pp. 297-310, Springer-Verlag, New York/Berlin 1989.

[Ba] L. Babai, "Trading Group Theory for Randomness", in Proceedings, 17th Annual ACM Symposium on the Theory of Computing, May 1985, pp 421-429.

[Bl] M. Blum, "Coin Flipping by Telephone", in Proceedings IEEE COMPCON 1982, pp. 133-137.

[BB] H. C. Bennett and G. Brassard, "Quantum cryptography: public key distribution and coin-tossing", in Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, Bangalore, India, pp. 175-179, 1984, quoted in [BCC].

[BBBSS] H.C. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin, "Experimental Quantum Cryptography", in Advances in Cryptology: Proceedings of EUROCRYPT '90, Aarhus, Denmark, Springer-Verlag,1991. Also submitted to Journal of Cryptology.

[BC1] G. Brassard and C. Crépeau, "Zero-Knowledge Simulation of Boolean Circuits", in Advances in Cryptology: Proceedings CRYPTO '86, Santa Barbara, CA, August 1986, pp. 223-233, Springer-Verlag, New York/Berlin, 1987

[BC2] G. Brassard and C. Crépeau, "Non-TransitiveTransfer of Confidence: A *Perfect* Zero-Knowledge Interactive Protocol for SAT and Beyond", in Proceedings, 27th Annual IEEE Symposium on the Foundations of Computer Science, October 1986, pp. 188-195.

[BC3] G. Brassard and C. Crépeau, "Quantum bit commitment and coin tossing protocols", in Advances in Cryptology: Proceedings CRYPTO '90, pp. 45-58, Springer-Verlag, to appear.

[BC4] G. Brassard and C. Crépeau, "Sorting out Zero-Knowledge ", in Advances in Cryptology: Proceedings EUROCRYPT '89, pp. 181-191, Springer-Verlag, 1990.

[BCC] G. Brassard, D. Chaum and C. Crépeau, "Minimum Disclosure Proofs of Knowledge", in Journal of Computer and System Sciences, Vol 37 No. 2, October 1988, pp. 156-189.

[BCR1] G. Brassard, C. Crépeau, and J.M. Robert, "All-or-Nothing disclosure of secrets", in Advances in Cryptology: Proceedings CRYPTO '86, Santa Barbara, CA, August 1986, pp. 235-238, Springer-Verlag, New York/Berlin, 1987.

[BCR2] G. Brassard, C. Crépeau, and J.M. Robert, "Information theoretic reductions among disclosure problems", in Proceedings, 27th Annual IEEE Symposium on the Foundations of Computer Science, October 1986, pp. 168-173.

[BCY] G. Brassard, C. Crépeau, and M. Yung, "Everything in NP can be argued in perfect zero-knowledge in a bounded number of rounds", Proceedings of the 16th International Colloquium on Automata, Languages and Programming, Springer-Verlag, 1989, pp. 123-136.

[BD] G. Brassard and I.B. Damgaard, "Practical IP $\subseteq$ MA", in Advances in Cryptology: Proceedings CRYPTO 88, pp. 580-582, Springer-Verlag, New York/Berlin, 1989.

[BFM] M. Blum, P. Feldman, and S. Micali, "Non interactive zero-knowledge and its applications", in 20th Annual ACM Symposium on the Theory of Computing, 1988, pp 103-112.

[BG] M. Bellare and S. Goldwasser, "New paradigms for digital signatures and message authentication based on non-interactive zero-knowledge proofs", in Advances in Cryptology: Proceedings CRYPTO '89, pp. 194-211, Springer-Verlag, 1990.

[BGG] M. Bellare, O. Goldreich, and S. Goldwasser, "Randomness in interactive proofs", in Proceedings, 31st Annual IEEE Symposium on the Foundations of Computer Science, 1990.

[BGGHKRM] M. Ben-Or, O. Golreich, S. Goldwasser, J. Hastad, J. Killian, P. Rogaway and S. Micali, "Everything Provable is Provable in Zero-Knowledge", in Advances in Cryptology: Proceedings CRYPTO 88, pp. 1-20, Springer-Verlag, New York/Berlin 1989.

[BGKW1] M. Ben-Or, S. Goldwasser, J. Killian, and A. Widgerson, "Multi-prover interactive proofs: How to remove intractability assumptions", in 20th Annual ACM Symposium on the Theory of Computing, 1988, pp 113-131.

[BGKW2] M. Ben-Or, S. Goldwasser, J. Killian, and A. Widgerson, "Efficient identification schemes using two prover interactive proofs", in Advances in Cryptology: Proceedings CRYPTO '89, pp. 459-469, Springer-Verlag, 1990.

[BKK] J.F Boyar, S.A. Kurtz, and M.W. Krentel, "A discrete logarithm implementation of zero-knowledge blobs", in Journal of Cryptology (1990) 2:63-76, Springer International.

[BM] M. Bellare and S. Micali, "Non interactive oblivious transfer and applications", in Advances in Cryptology: Proceedings CRYPTO '89, pp. 547-557, Springer-Verlag, 1990.

[BMO] M. Bellare, S. Micali and R. Ostrovsky, "Perfect Zero-knowledge in constant rounds" in 22nd Annual ACM Symposium on the Theory of Computing, 1990, pp 482-493.

[BP] J. Boyar and R. Peralta, "On the concrete complexity of zero-knowledge proofs", in Advances in Cryptology: Proceedings CRYPTO '89, pp. 507-525, Springer-Verlag, 1990.

[BY] G. Brassard and M. Yung, "One way group actions", in Advances in Cryptology: Proceedings CRYPTO '90, Pp. 85-98, Springer-Verlag, 1991.

[C] S.A. Cook, "The complexity of Theorem Proving Procedures", in Proceedings 3rd Annual ACM Symposium on the Theory of Computing, 1971, pp 151 - 158.

[Ch] D. Chaum, "Demonstrating that a Public Predicate can be Satisfied Without Revealing any Information About How", in Proceedings CRYPTO 86, pp 195 - 199, Springer Verlag, 1987

[Co] J. Cohen (Benaloh), "Cryptographic Capsules: A disjunctive Primitive for Interactive Protocols", in Advances in Crypology: Proceedings CRYPTO 86, pp. 213-222, Springer-Verlag 1987.

[Cr] C. Crepeau, "Equivalence between two flavours of oblivious transfers", in Advances in Cryptology: Proceedings, CRYPTO '87, Santa Barbara, CA, August 1987, pp. 350-354, Springer-Verlag, New York/Berlin, 1988.

[CDG] D. Chaum, I. B. Damgard, and J. Van de Graaf, "Multiparty Computations ensuring privacy of each party's input and correctness of the result", in Advances in Cryptology: Proceedings, CRYPTO '87, Santa Barbara, CA, August 1987, pp. 87-119, Springer-Verlag, New York/Berlin, 1988.

[CEGP] D. Chaum, J.H Evertse, J. Van de Graaf and R. Peralta, "Demonstrating Possession of a Discrete Logarithm without Revealing it", in Proceedings CRYPTO 86, pp. 200-212, Springer-Verlag, 1987.

[CEG] D. Chaum, J.H Evertse, and J. Van de Graaf, "An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some

Generalizations", in Proceedings EUROCRYPT 87 , pp. 127-141, Springer-Verlag, 1988.

[CFN] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash", in Advances in Cryptology: Proceedings, CRYPTO '88, pp 319-327, Springer-Verlag, New York/Berlin, 1989.

[CV] D. Chaum and H. van Antwerpen, "Undeniable Signatures", in Advances in Cryptology: Proceedings CRYPTO '89, pp. 212-216, Springer Verlag, 1990.

[D] I.V. Damgard, " On the existence of bit commitment schemes and zero-knowledge proofs", in Advances in Cryptology: Proceedings CRYPTO '89, pp. 18-27, Springer Verlag, 1990.

[De] D. Denning, "Cryptography and Data Security", Addison-Wesley Publishing Company, Inc, 1982.

[DGB] Y. Desmedt, C. Goutier, and S. Bengio, "Special uses and abuses of the Fiat-Shamir passport protocol", in Advances in Cryptology: Proceedings CRYPTO '87, pp. 21-39, Springer Verlag, 1988.

[DMP1] A. De Santis, S. Micali, and G. Persiano, "Non interactive zero-knowledge proof systems", in Advances in Cryptology: Proceedings CRYPTO '87, pp. 52-72, Springer Verlag, 1988.

[DMP2] A. De Santis, S. Micali, and G. Persiano, "Non interactive zero-knowledge with preprocessing", in Advances in Cryptology: Proceedings CRYPTO '88, pp. 269-282, Springer Verlag, 1989.

[DY] A. De Santis and M. Yung, "Cryptographic applications of the non interactive metaproof and many-prover systems", in Advances in Cryptology: Proceedings CRYPTO '90, pp. 357-369, Springer Verlag, to appear.

[F] L. Fortnow, "The Complexity of Perfect Zero-Knowledge", in Proceedings, 19th Annual ACM Symposium on the Theory of Computing, May 1987, pp 204-209.

[FeS1] U. Feige and A. Shamir, "Zero knowledge proofs of knowledge in two rounds", in Advances in Cryptology: Proceedings CRYPTO '89, pp. 526-544, Springer Verlag, 1990.

[FeS2] U. Feige and A. Shamir, "Witness hiding protocols" in Proceedings 22nd Annual ACM Symposium on the Theory of Computing, May 1990, pp. 416-426.

[FiS] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to Identification and Signature problems", in Advances in Cryptology: Proceedings CRYPTO 86, pp. 186-194, Springer-Verlag, 1987.

[FFS] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity", in Proceedings, 19th Annual ACM Symposium on the Theory of Computing, May 1987, pp. 210-217.

[FLS] U. Feige, D. Lapidot, and A. Shamir, "Multiple non interactive zero-knowledge proofs based on a single random string", in Proceedings, 31st Annual IEEE Symposium on the Foundations of Computer Science, 1990, quoted in [LS].

[GHY] Z. Galil, S. Haber and M. Yung, "A private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public-Key Cryptosystems", in Proceedings, 26th Annual IEEE Symposium on the Foundations of Computer Science, 1985, pp. 360-371.

[GJ] M.R. Garey and D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", Freeman, New York 1979.

[GL] O. Goldreich and L. Levin, "A hard-core predicate for all one-way functions", in proceedings 21st Annual ACM Symposium on the Theory of Computing, May 1989, pp 25-32.

[GM] S. Goldwasser and S. Micali, "Probabilistic Encryption", in Journal of Computer and System Sciences, Vol 28, 1984, pp. 270-299.

[GMR1] S. Goldwasser, S. Micali and C. Rackhoff, " The knowledge Complexity of Interactive Proof Systems", in 17th Annual ACM Symposium on the Theory of Computing, May 1985, pp 291-304.

[GMR2] S. Goldwasser, S. Micali and C. Rackhoff, " The knowledge Complexity of Interactive Proof Systems", in Siam J. Computing Vol 18, No.1, pp. 186-208, February 1989

[GMW1] O. Goldreich, S. Micali and A. Widgerson, "Proofs that Yield Nothing but their Validity and a Methodology of Cryptographic Protocol Design", in Proceedings, 27th Annual IEEE Symposium on the Foundations of Computer Science, 1986, pp. 174-187.

[GMW2] O. Goldreich, S. Micali and A. Widgerson, "How to Prove all NP Statements in Zero-Knowledge and a Methodology of Cryptographic Protocol Design", in Advances in Cryptology: Proceedings CRYPTO 86, pp. 172-185, Springer-Verlag, 1987.

[GQ1] L.C. Quillou and J.J. Quisquater, "A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory" in Advances in Cryptology: Proceedings, EUROCRYPT '88, pp 123-128, Springer-Verlag, New York/Berlin, 1989.

[GQ2] L.C. Quillou and J.J. Quisquater, "A 'paradoxical' identity-based signature scheme resulting from zero-knowledge" in Advances in Cryptology: Proceedings, CRYPTO '88, pp 216-231, Springer-Verlag, New York/Berlin, 1989.

[GQ3] L.C. Quillou and J.J. Quisquater, "How to explain zero-knowledge protocols to your children", in Advances in Cryptology: Proceedings CRYPTO '89, Springer Verlag, 1990.

[GS] S. Goldwasser and M. Sipser, "Private Coins Versus Public Coins in Interactive Proofs Systems", in Proceedings, 18th Annual ACM Symposium on the Theory of Computing, 1986, pp 59-68.

[IL] R. Impagliazzo and M. Luby, "One-way functions are essential for complexity based cryptography", in Proceedings, 30th Annual IEEE Symposium on the Foundations of Computer Science, 1989, pp. 230-235.

[ILL] R, Impagliazzo, L. Levin, and M. Luby, "Pseudo-random generation from one-way functions", in proceedings 21st Annual ACM Symposium on the Theory of Computing, May 1989, pp 12-24.

[IY] R. Impagliazzo and M. Yung, "Direct Minimum-Knowledge Computations", in Advances in Cryptology: Proceeedings CRYPTO '87, pp. 40-51, Springer Verlag, 1988.

[K] J. Killian, "Founding Cryptography on oblivious transfer", in Proceedings, 20th Annual ACM Symposium on the Theory of Computing, 1988, pp 20-31.

[Kn] H.J. Knobloch, "A smart card implementation of the Fiat-Shamir identification scheme", in Advances in Cryptology: Proceedings, EUROCRYPT '88, pp 87-95, Springer-Verlag, New York/Berlin, 1989.

[KMO] J. Killian, S. Micali, and R. Ostrovsky, "Minimum resource zero-knowledge proofs", in Proceedings, 30th Annual IEEE Symposium on the Foundations of Computer Science, 1989, pp. 474-479. An earlier version appears in CRYPTO 89, pp. 545-546, Springer-Verlag,1990.

[LS] D. Lapidot and A. Shamir, "Publicly verifiable non-interactive zero-knowledge proofs", in Advances in Cryptology: Proceedings CRYPTO '90, Pp. 339-356, Springer-Verlag, 1991.

[MS] S. Micali and A. Shamir, "An improvement of the Fiat-Shamir identification and signature scheme", in Advances in Cryptology: Proceedings CRYPTO '88, pp. 244-247, Springer Verlag, 1989.

[N] M. Naor, "Bit commitment using pseudo-randomness", in Advances in Cryptology: Proceedings CRYPTO '89, pp. 128-136, Springer-Verlag, New York/Berlin 1990.

[NY] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks", in Proceedings, 22nd Annual ACM Symposium on the Theory of Computing, 1990, pp. 427-437.

[O] Y. Oren, "On the Cunning Power of Cheating Verifiers: Some Observations of Zero-Knowledge Proofs", in Proceedings, 28th Annual IEEE Symposium on the Foundations of Computer Science, October 1987, pp. 462-471.

[Od] A. M. Odlyzko, "Discrete logarithms in finite fields and their cryptographic significance", Proceedings, EUROCRYPT '84, Berlin, Heidelberg 1984,pp. 224-260, Springer-Verlag, 1985.

[OO] K. Ohta and T. Okamoto, "A modification of the Fiat-Shamir scheme", in Advances in Cryptology: Proceedings CRYPTO '88, pp. 232-243, Springer Verlag, 1989.

[Pe] R. Peralta, "Simultaneous security of bits in the discrete log", in Advances in Cryptology: Proceedings, EUROCRYPT '85, Linz, Austria, April 1985, pp 62-72, Springer-Verlag, New York/Berlin, 1986.

[PG] R. Peralta and J. Van de Graaf, "A simple way to show the validity of your public key", in Advances in Cryptology: Proceedings, CRYPTO '87, Santa Barbara, CA, August 1987, pp 128-134, Springer-Verlag, New York/Berlin, 1988.

[TW] M. Tompa and H. Woll, "Random Self-Reducibility and Zero-knowledge Interactive Proofs of Possession of Information", in Proceedings, 28th Annual IEEE Symposium on the Foundations of Computer Science, October 1987, pp. 472-482.